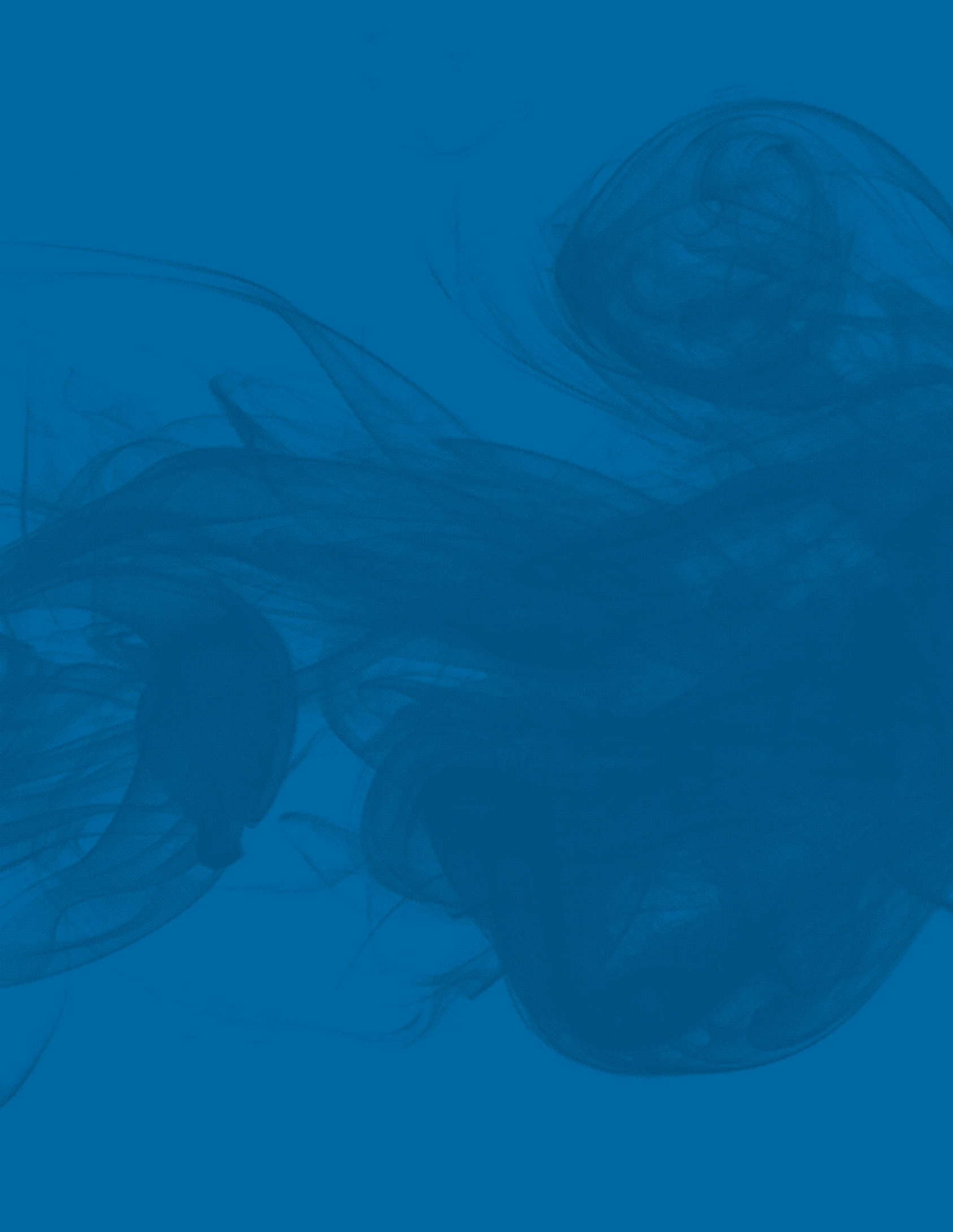
FEATURING

- THE BEST PLUGINS
- CREATE & MAINTAIN PERFECT THEMES
- RESPONSIVE DESIGN TECHNIQUES
- BUILD FLEXIBLE LAYOUTS
- GET STARTED WITH THE REST API
- CREATE BEAUTIFUL PORTFOLIOS
- SPEED UP YOUR SITE
- DISCOVER THE GENESIS FRAMEWORK
- MOVE FROM CSS TO SASS
- ADVANCED CUSTOM FIELDS TIPS & TRICKS
- AND MUCH MORE!

THE ULTIMATE GUIDE TO WORDPRESS VOLUME II

148 PAGES OF EXPERT WORDPRESS TIPS, TUTORIALS & TECHNIQUES

Future From the makers of net



COLOPHON

@netmag /netmag +netmagazine *flickr.com/photos/netmag*
netmag@futurenet.com *net.creativebloq.com*

Future PLC, Quay House, The Ambury, Bath, BA1 1UA +44 (0)1225 442244

EDITORIAL

Editor **Oliver Lindberg** *oliver.lindberg@futurenet.com*
Production editor **Ruth Hamilton** *ruth.hamilton@futurenet.com*
Art editor **Mike Brennan** *mike.brennan@futurenet.com*

EDITORIAL CONTRIBUTIONS

Kirsty Burgoine, Mike Brown, Joe Casabona, Kim Crawley, Ian Devlin, Carrie Dils, Corey Ellis, Zac Gordon, Cole Henley, Sam Hernandez, Sam Kapila, Joseph Karr O'Connor, James Koster, Mark Llobrera, Jenn Lukas, Eric Mann, Ryan McCue, Rachel McCollin, Cassie McDaniel, Tim Nash, Yesenia Perez-Cruz, Josh Pollock, Tobias van Schieider, Shannon Smith, James Steinbach, Ryan Taylor, Kat Thompson, Amber Weinberg

ART CONTRIBUTIONS

Alexandra Bruel, Maricor/Maricar, Bex Shaw

MANAGEMENT

Content and marketing director **Nial Ferguson** *nial.ferguson@futurenet.com*
Head of content & marketing, photography, creative and design
Matthew Pierce *matthew.pierce@futurenet.com*
Group editor-in-chief **Dan Oliver** *dan.oliver@futurenet.com,*
Group art director **Rodney Dive** *rodney.dive@futurenet.com*
Group content manager, creative and design
Tom May *tom.may@futurenet.com*

ADVERTISING Advertising manager
Sasha McGregor sasha.mcgregor@futurenet.com
CIRCULATION Trade marketing manager
Juliette Winyard *juliette.winyard@futurenet.com*
PRODUCTION Production controller **Nola Cokely** *nola.cokely@futurenet.com*
Production manager **Mark Constance** *mark.constance@futurenet.com*
LICENSING International director **Regina Erak** *regina.erak@futurenet.com*
SUBSCRIPTIONS Phone our UK hotline 0844 848 2852; international +44 (0)1604 251 045 Subscribe to **net** online at *myfavouritemagazines.co.uk*

Future is an award-winning international media group and leading digital business. We reach more than 49 million international consumers a month and create world-class content and advertising solutions for passionate consumers online, on tablet & smartphone and in print.

Future plc is a public company quoted on the London Stock Exchange (symbol: FUTR) www.futureplc.com

Chief executive Zillah Byng-Maddick
Non-executive chairman Peter Allen
Chief financial officer Richard Haley

Tel +44 (0)207 042 4000 (London)
Tel +44 (0)1225 442 244 (Bath)

FSC www.fsc.org MIX Paper from responsible sources FSC® C007184

recycle When you have finished with this magazine please recycle it.

We are committed to only using magazine paper which is derived from well managed, certified forestry and chlorine-free manufacture. Future Publishing and its paper suppliers have been independently certified in accordance with the rules of the FSC (Forest Stewardship Council).

INTRODUCTION

Welcome to The Ultimate Guide to WordPress! In this special from the makers of **net** magazine, we bring you everything you need to know to get ahead with the world's most popular content management system.

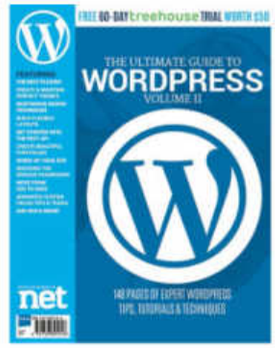
We'll kick off with offering you some inspiration through a showcase of some of the best WordPress sites around, followed by some introductory features to get you started. Next, there's a massive 78-page practical section, packed with tutorials on everything from theming and making your WordPress site responsive to moving from CSS to Sass and building with the WordPress REST API. Finally, there's a resources section with the best blog themes, plugins and tutorials you should check out.

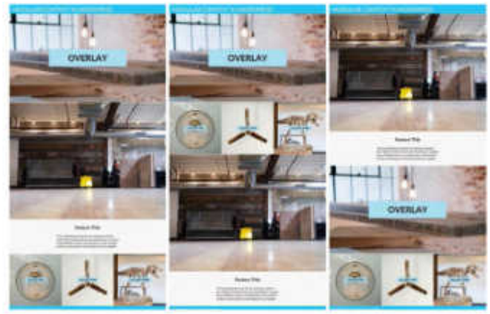
We've also teamed up with Treehouse (whose WordPress expert Zac Gordon presents a self-guided WordPress curriculum on page 42) to offer you an exclusive 60-day trial (worth $50). This will provide you with full access to more than 1,000 online videos, not only on WordPress but also on all other aspects of web design, coding, business and more! Simply visit *jointreehouse/netmag* to sign up.

Enjoy!

Oliver Lindberg, editor
oliver.lindberg@futurenet.com
@oliverlindberg

CONTENTS


THE ULTIMATE GUIDE TO
WORDPRESS
VOLUME II
net


OVERLAY
OVERLAY
OVERLAY


WP-Portfolio


Genesis Framework
Sample Post With Threaded Comments

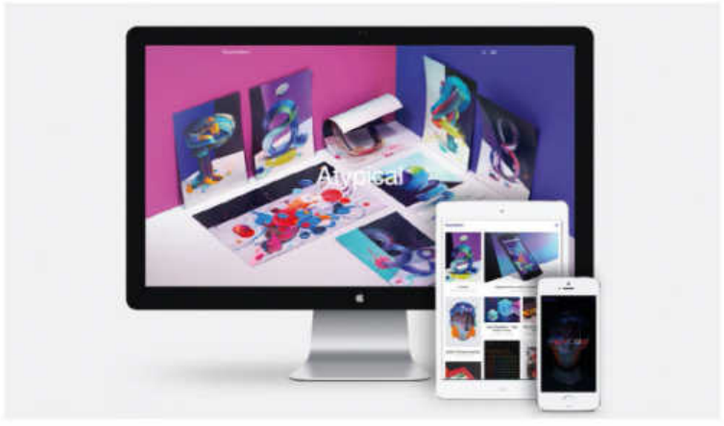

Building Parsec
Hello world!

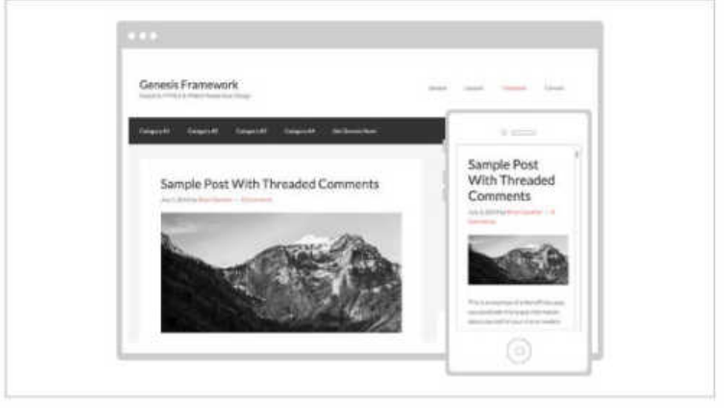

30
MUST-HAVE
WORDPRESS
PLUGINS

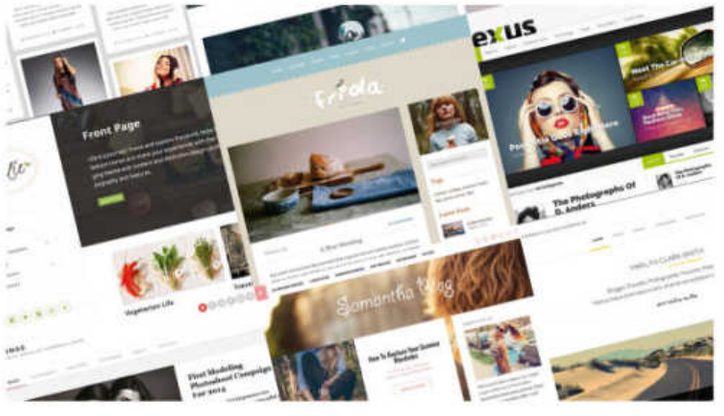

MARIO BATALI


HANDCRAFTED

Contents







SHOWCASE

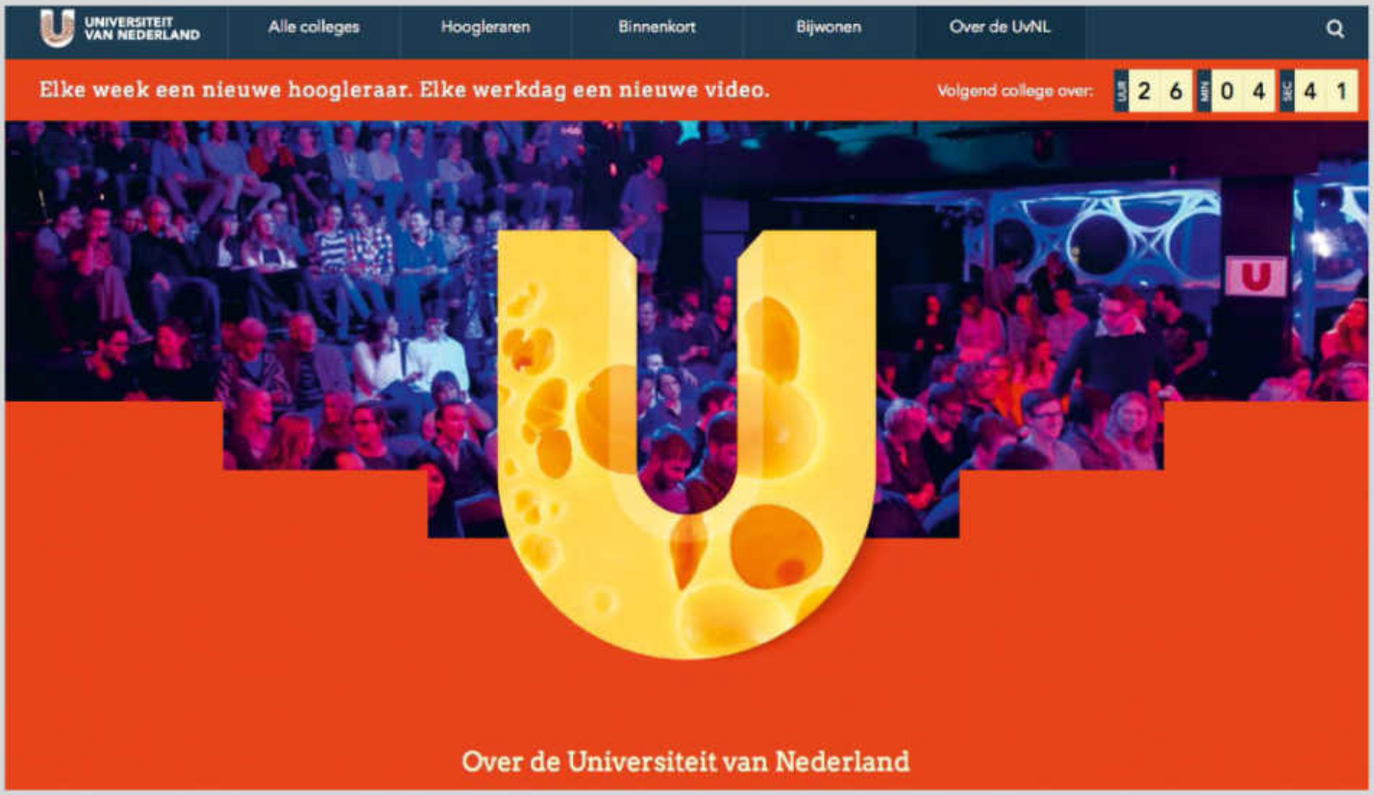
GETTING STARTED

PROJECTS

RESOURCES

GALLERY

Sensational design and superb development



TYPOGRAPHY, WORDPRESS, HTML5

UNIVERSITEITVANNEDERLAND.NL

Jort de Vries *jortdevries.nl* (designer), **Maikel van der Zander** (developer)

As somebody who teaches web design at the University of Greenwich, it is refreshing to see an education site that has usability, inclusivity and strong design at its core. The aim of the site is to bring lectures from university professors to a broader audience through accessible, digestible videos.

"We offer those professors a chance to share their college teaching with everyone in the Netherlands – 16 million students," explains designer Jort de Vries.

The site is built with responsiveness in mind, focusing on a clear content hierarchy, reinforced through strong typography and a bold use of colour. This is supported by a flexible identity, using a simple, iconic motif and complementary imagery to illustrate the breadth of subjects covered. Videos are hosted and delivered by YouTube, using HTML5 to integrate the video content within the site design.

WORDS: Cole Henley

"Clearly organised, with beautiful typography. Better yet, it's responsive. Universities: this is how it's done"
CHRISTOPHER MURPHY (@FEHLER)

WORDPRESS, APIS

TRAVELPORTLAND.COM

Switchyard Creative *switchyardcreative.com*

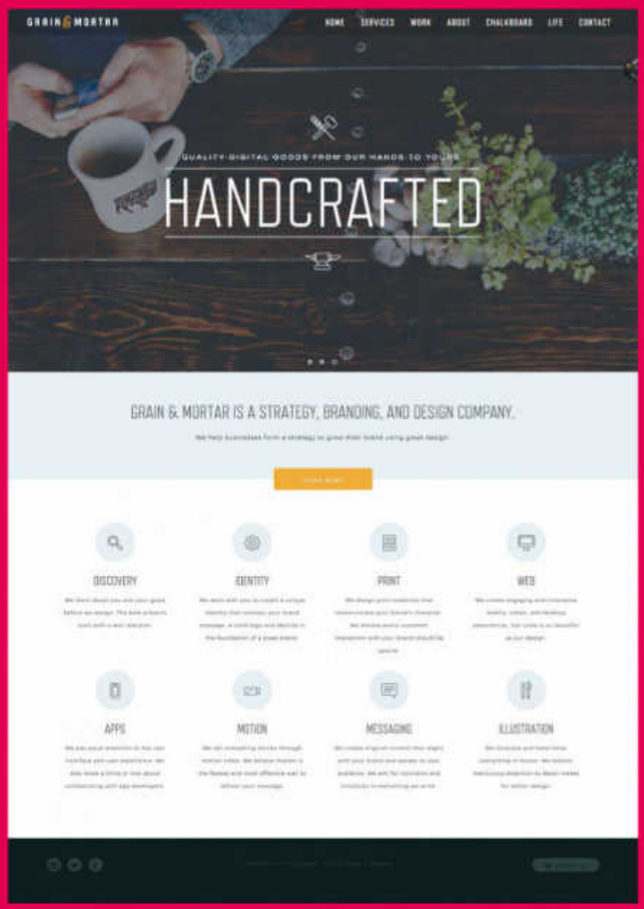
Travel Portland aims to inspire people to visit Portland in Oregon, and to help them plan their trip with tools and content highlighting the unique opportunities in the city. The site design is clean and clear, offering up images that capture the Portland experience, from waterfalls to food trucks.

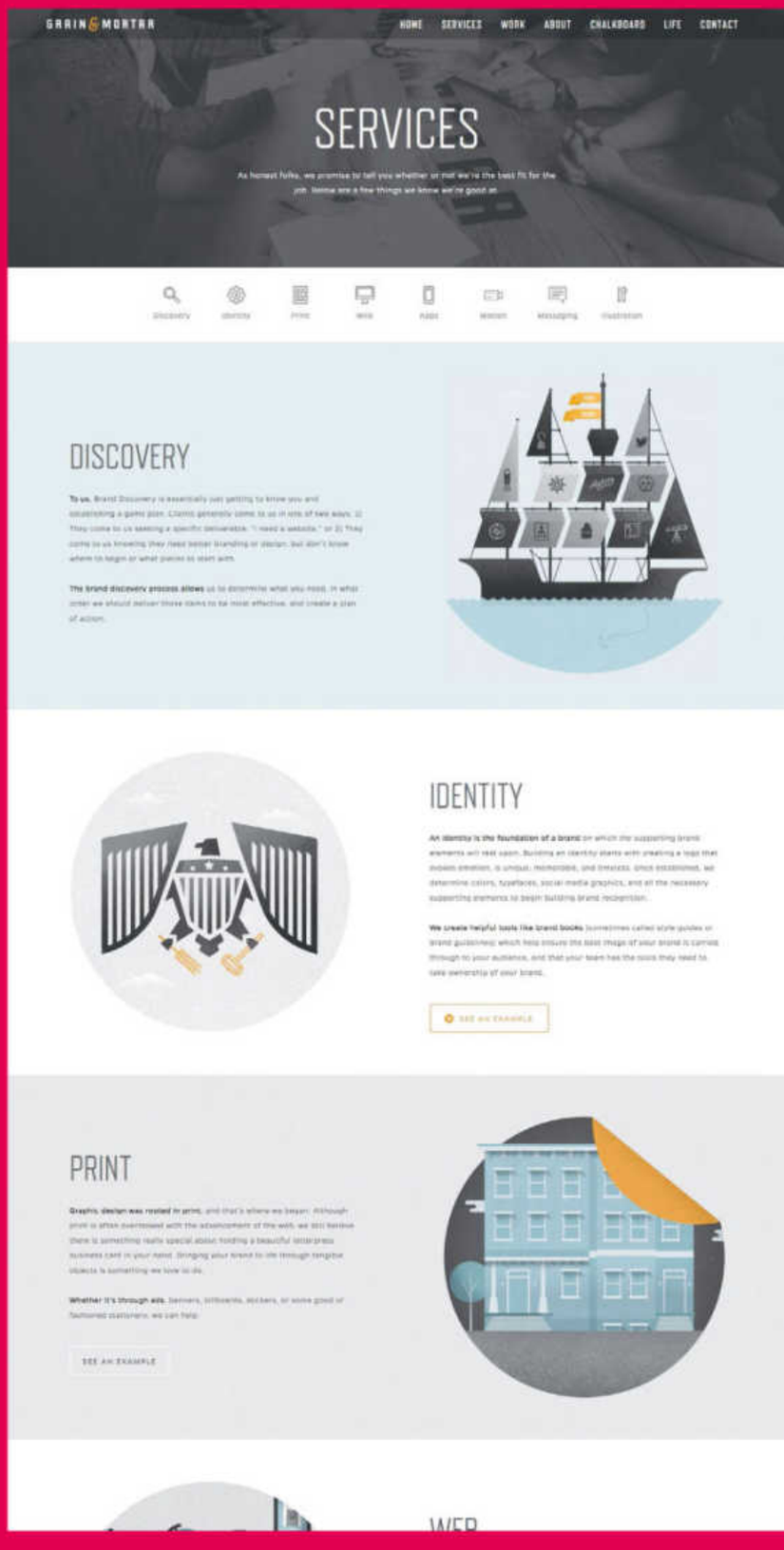
"We placed a strong emphasis on photography to convey the beauty of the city and the region, with the images serving double duty as page content and navigational elements," says Richard Tammar, director of online strategy for Travel Portland. "Mid-production we made some tweaks to serve our highest resolution images to Retina displays."

The beautifully responsive site is built on WordPress and uses a customised responsive theme based on Zurb's Foundation framework. Third-party APIs are utilised throughout, including Storify, MapBox, Weather Underground and ChooseCulture.

The team used MaxMind's geolocation API to give them the ability to customise content on the homepage based on the location of the visitor. This lets them serve different information for in-town visitors versus those in the planning phases. Tammar adds, "Our weather widget will let you know if it's nicer in Portland than where you're browsing from – which it often is, by the way!"

WORDS: Jenn Lukas





ILLUSTRATIONS, RWD, WORDPRESS, HTML5 VIDEO

GRAINANDMORTAR.COM

Grain & Mortar *grainandmortar.com*

Grain & Mortar is a strategy, branding and design company based in Omaha, Nebraska. For its recent redesign it has aimed for a gritty versus polished-and-perfect look, with clean typeface and strong lines contrasting with fun, textured illustrations and ornate headings. The studio members are featured throughout the site, so you get a real feel for the company and the people just by navigating though the pages.

"We wanted to use photography and illustration to create more of a lifestyle feel," explains the company's operations editor Kristin DeKay. "One of our favourite new pages – *grainandmortar.com/life* – is an aggregator of all our posts from Instagram and Dribbble, so visitors can see what we've been up to recently."

The site uses WordPress as the CMS and takes advantage of HTML5 video and subtle CSS3 animations to enhance the user experience. It's also beautifully responsive with attention being paid to every detail. Go on, shrink that browser! Finally if (like me) you're a sucker for a trendy office space, make sure you check out the About page.

WORDS: Ryan Taylor

"Illustrations take it up a notch. I love the textures and details – gritty and handmade, like the name implies"

GERI COADY
@HELLOGERI

RESPONSIVE, HTML5, WORDPRESS

WORRYFREELABS.COM

Worry Free Labs *worryfreelabs.com*

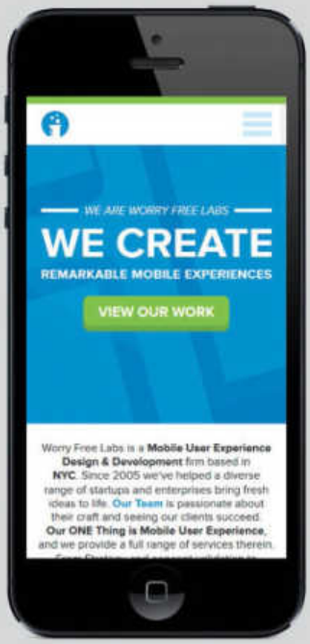
Worry Free Labs is a web agency based in New York that prides itself on building remarkable mobile experiences. Where better to show off its skills than on its own site?

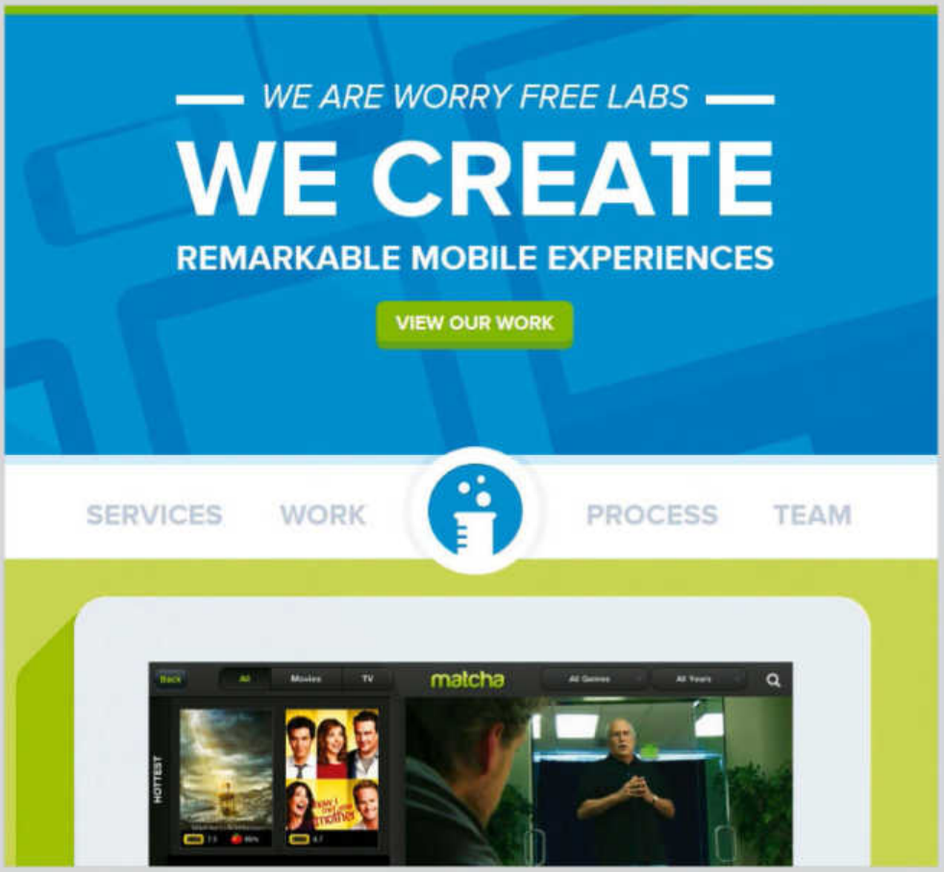
Jason Curry, founder of Worry Free Labs, explains that, when monitoring the website traffic the team had seen a dramatic increase in visitors using mobile and tablet devices. A responsive website was clearly the way to go.

Like most web agencies, Worry Free Labs decided to make use of HTML5. "It's clean, concise and effective," says Curry. "But we prefer it mostly for being alive, changing and up-to-date. It's evolving just like a human language does."

There are some lovely touches throughout the site that make browsing on mobile an enjoyable experience. There's easy sliding and animation on the Work page, and the automated reveal on scroll slides under Services, and a useful 'Swipe to Send' button on the contact form.

WORDS: Ian Devlin





WORDPRESS, ANGULARJS, ANIMATION

HARUKIMURAKAMI.COM

Bluecadet *bluecadet.com*

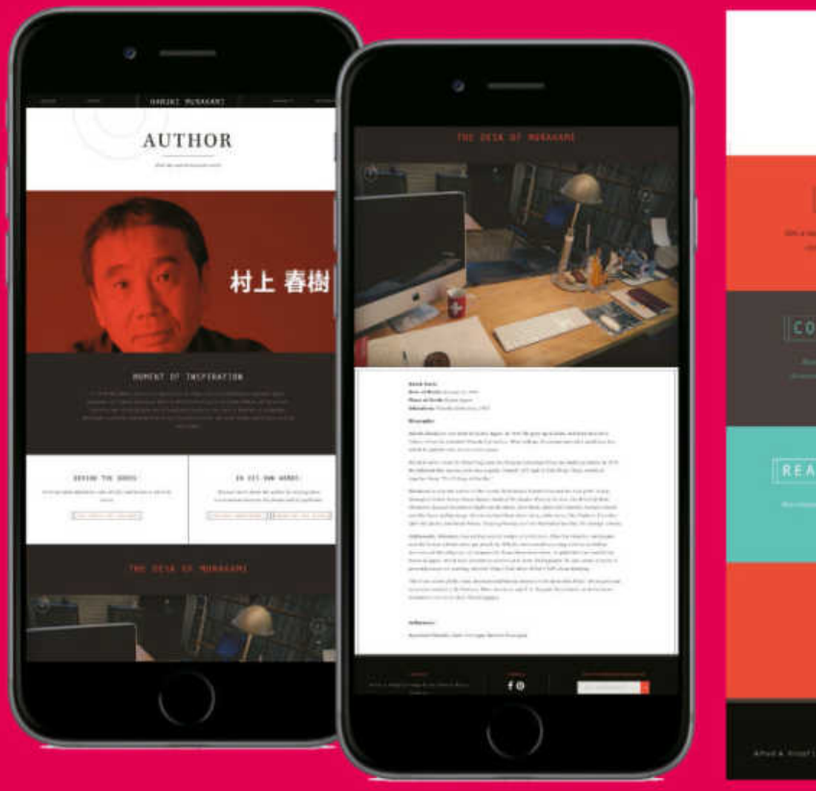
Japanese author Haruki Murakami is known for novels that deal with duality, conflict, fantasy and reality, played out through multiple plot lines. His new website, designed by Bluecadet in collaboration with Knopf Doubleday Publishing Group, captures many of his novels' recurring themes.

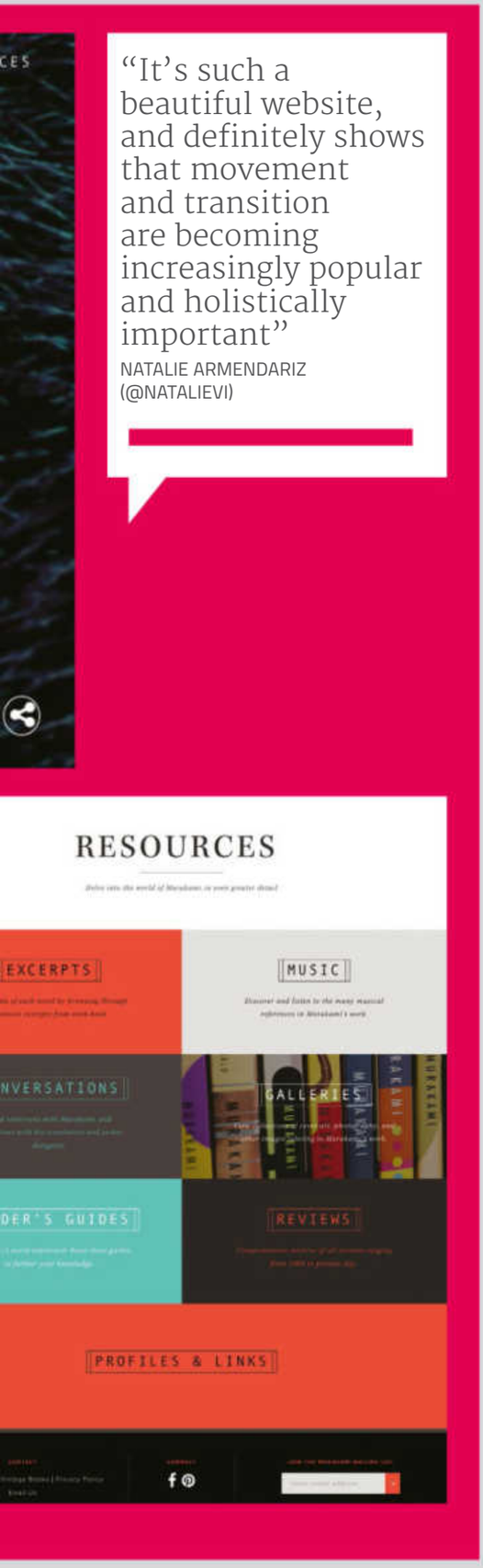
The Library section of the site displays a selection of mesmerising covers of Murakami's novels. It makes use of subtle animation powered by AngularJS on hover states and in the background, which seduces the viewer into complete aesthetic bliss. The About section highlights what appears to be Murakami's desk, with clickable light-boxed photos of desk details captioned in the first person.

The site includes a Community page, where visitors can share and read other peoples' experiences of the author's books. As the page loads, the elements animate to give the impression of a bookstore filling up with a crowd attending a book reading.

Kepler Std, Andale Mono and Proxima Nova, served by Typekit and `@font-face`, look lovely together and help carry the theme of duality through the site. They work particularly well in the Resources section, where they lend a delicate and elegant aesthetic to the reading guides, reviews and conversations of Murakami's work.

WORDS: Sam Kapila





"It's such a beautiful website, and definitely shows that movement and transition are becoming increasingly popular and holistically important"

NATALIE ARMENDARIZ (@NATALIEVI)

MAGENTO, WORDPRESS, CSS3 ANIMATION, FONTFACE

ALTERNATIVEAPPAREL.COM

Instrument *instrument.com*

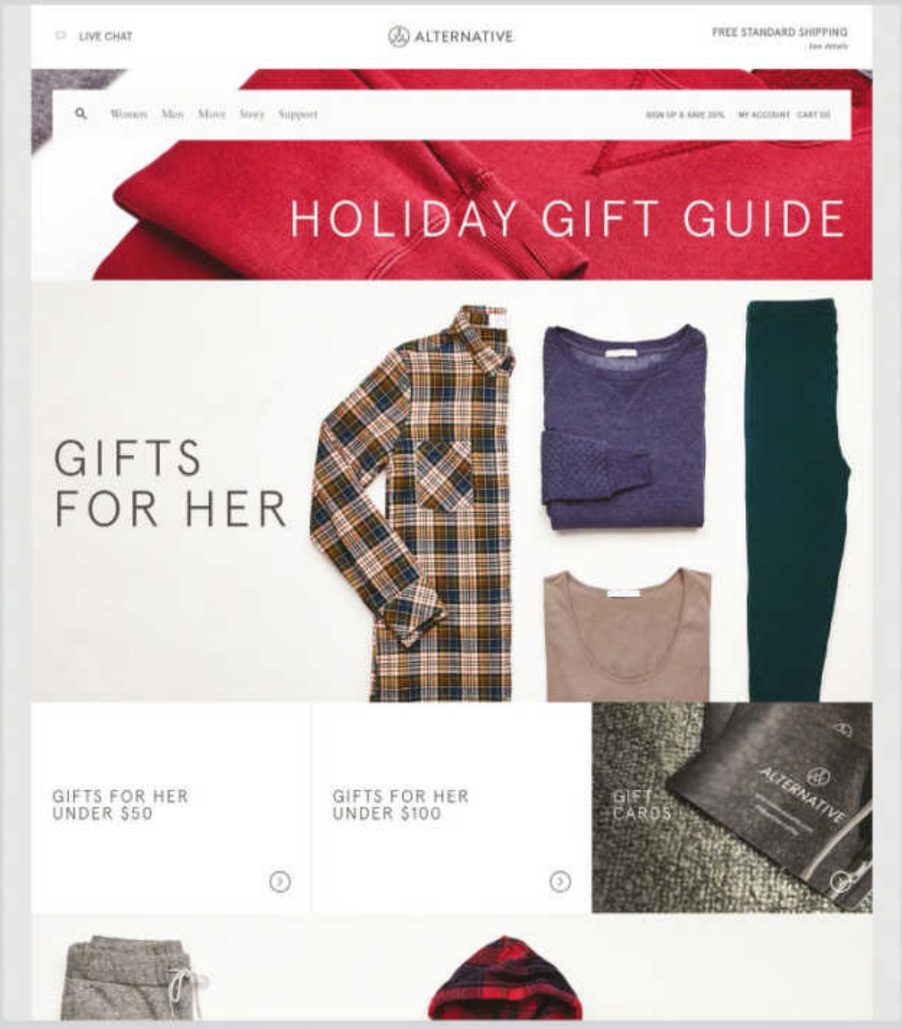
Alternative Apparel is a fully responsive ecommerce site, designed by Portland-based agency Instrument. To ensure maximum flexibility, the site was built using a mix of two platforms: WordPress for building and managing content, and Magento for displaying it.

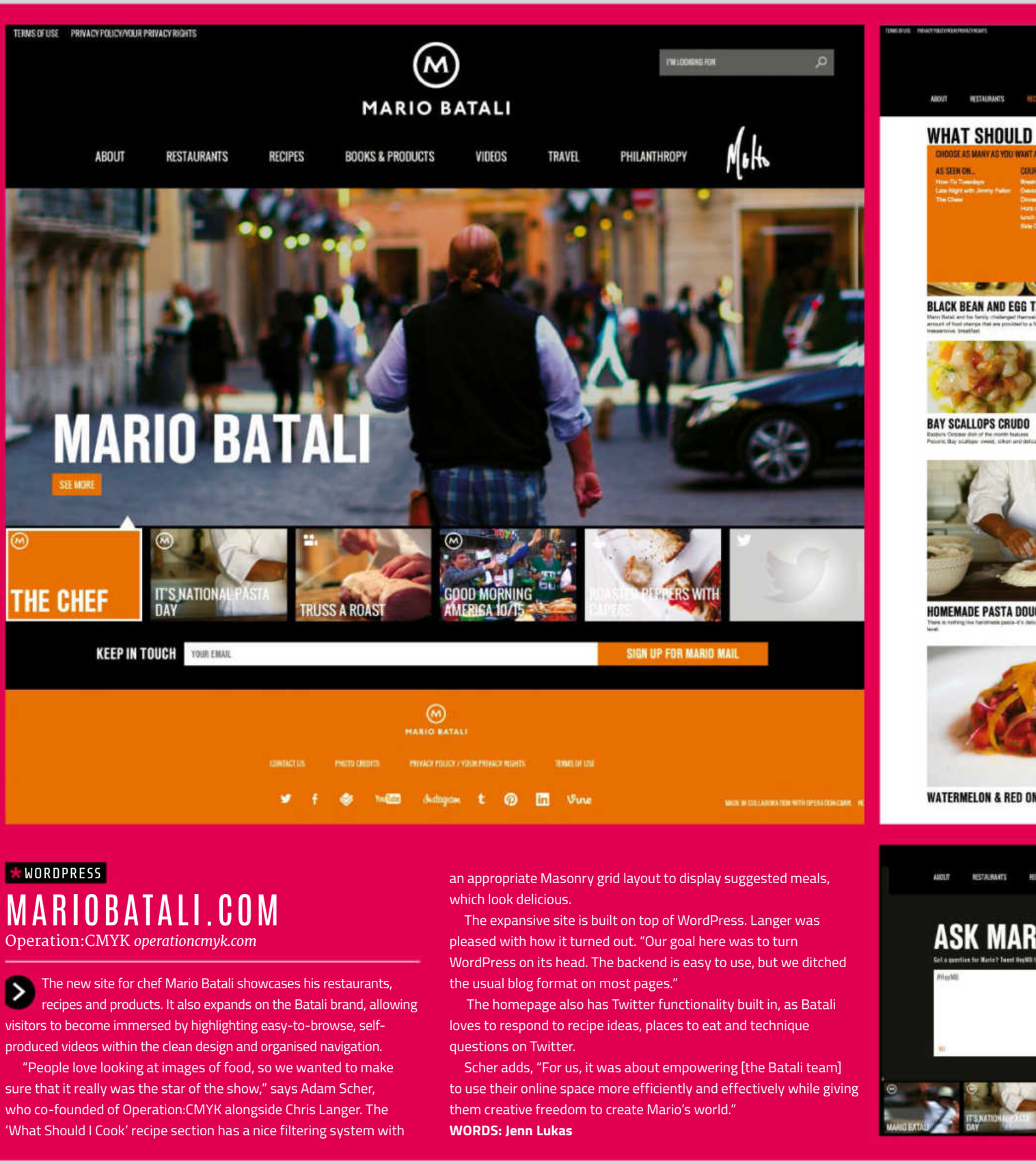
"The CMS functionality offered by Magento is fairly minimal and quite complicated for marketers or content editors. WordPress, on the other hand, is well known for being easy to grasp," says senior developer Jim Dalton.

Instrument used WordPress to build a plugin that added custom post types that would map to content types required in Magento. It also adapted the admin side to make it easier for content editors to work with.

"The client was very pleased with this integration," Dalton continues. "The extra mile we went in making the content creation and layout process seamless has empowered their content creation and marketing teams. It's gratifying to see how frequently their merchandising team is changing up the content, which wouldn't have been nearly as practical if we had relied solely on Magento's out-of-the-box CMS functionality."

WORDS: Yesenia Perez-Cruz





WORDPRESS

MARIOBATALI.COM

Operation:CMYK *operationcmyk.com*

The new site for chef Mario Batali showcases his restaurants, recipes and products. It also expands on the Batali brand, allowing visitors to become immersed by highlighting easy-to-browse, self-produced videos within the clean design and organised navigation.

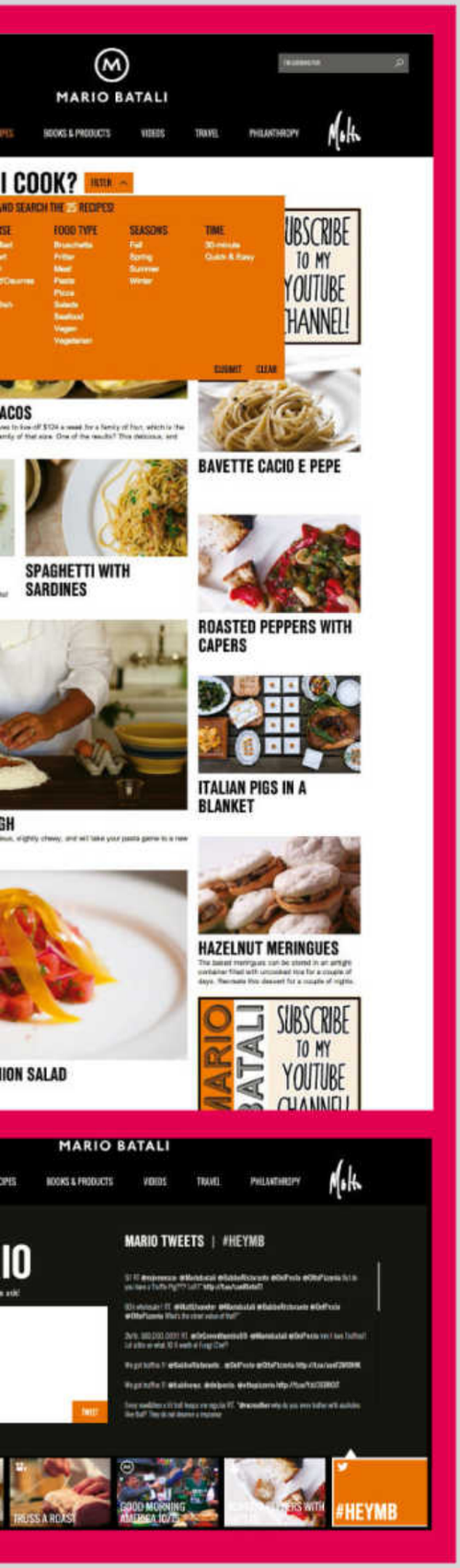
"People love looking at images of food, so we wanted to make sure that it really was the star of the show," says Adam Scher, who co-founded of Operation:CMYK alongside Chris Langer. The 'What Should I Cook' recipe section has a nice filtering system with an appropriate Masonry grid layout to display suggested meals, which look delicious.

The expansive site is built on top of WordPress. Langer was pleased with how it turned out. "Our goal here was to turn WordPress on its head. The backend is easy to use, but we ditched the usual blog format on most pages."

The homepage also has Twitter functionality built in, as Batali loves to respond to recipe ideas, places to eat and technique questions on Twitter.

Scher adds, "For us, it was about empowering [the Batali team] to use their online space more efficiently and effectively while giving them creative freedom to create Mario's world."

WORDS: Jenn Lukas



WORDPRESS

THISISYOURKINGDOM.CO.UK

Katie Marcus *whatkatiedoes.net*
Kim Lawler *www.kimlawlercreative.com*

This is Your Kingdom is a site that brings together style and substance to help visitors find activities in their local area. Simple in its concept, the site is much more complex when we start to consider all of its inner workings and potential.

Designed by Katie Marcus, and robustly developed by Kim Lawler, the site is built to handle growing levels of content. Lawler used WordPress to allow site editors and guest bloggers to update easily and regularly, giving visitors plenty of reasons to return regularly.

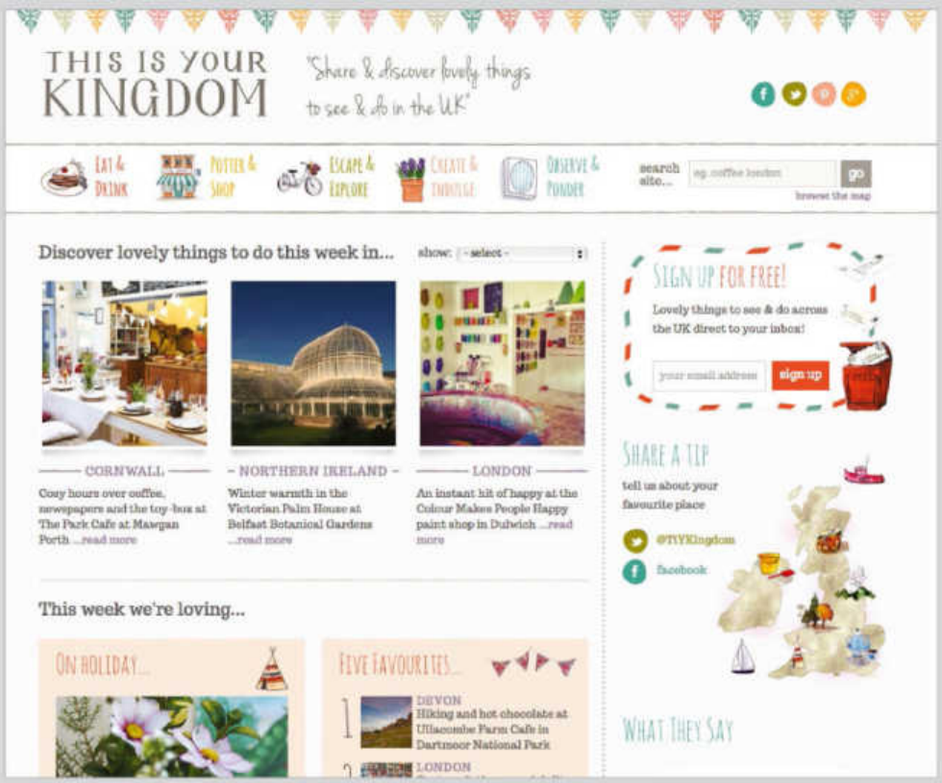
Under the hood, the beautiful work continues. Lawler says, "There were several challenges, not least the organisation of the articles. Custom taxonomies play a huge part in making the content less linear and create a richer browsing experience for site visitors."

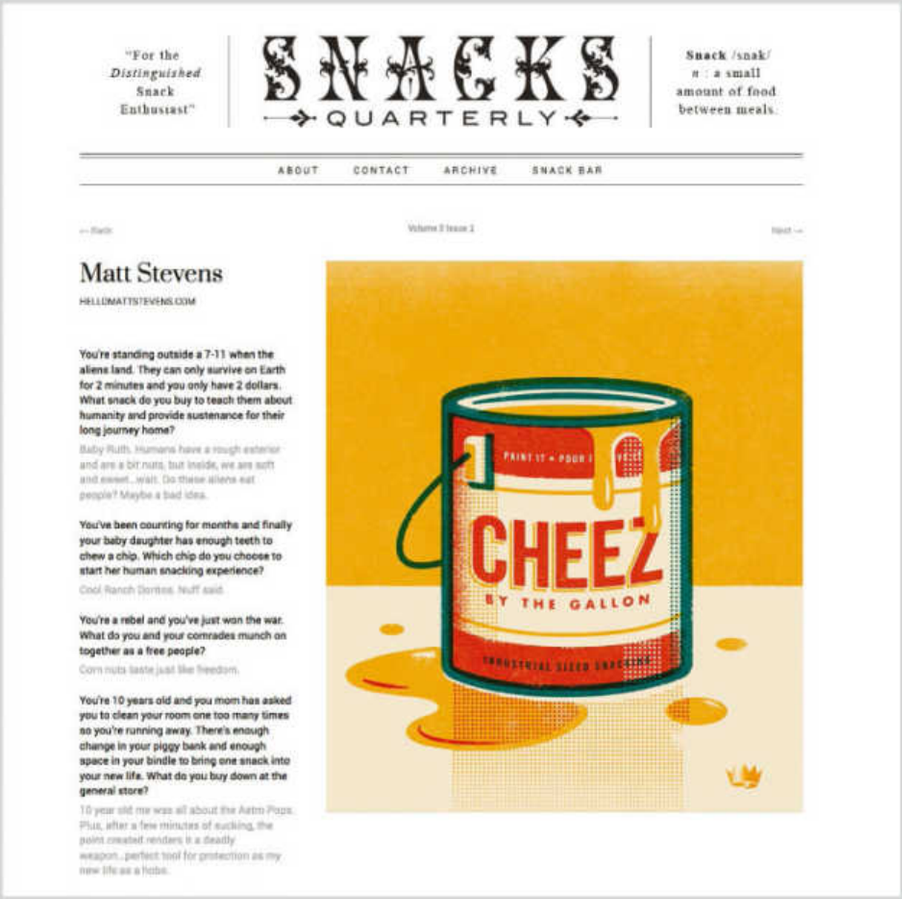
The delicate, playful design hints at a wide range of activities shared on the site, and does a great job of being appealing, without being stereotypically girly or niche. Illustration plays a big part in the appeal of the site, paving the way for imagination and exploration.

Lawler adds, "This is Your Kingdom is probably one of my proudest WordPress builds." It's easy to see why.

WORDS: Kat Thompson

> "TiYK has made a stylish, usable site that I kinda wish I'd made"
> LAUREN THOMPSON (@LRNN)





HTML5, WORDPRESS, JQUERY

SNACKSQUARTERLY.COM

Alexander Barrett *alexanderbarrett.com*, **Brad Simon** *bradsimonart.com*

When I discovered Snacks Quarterly I loved it immediately. Unabashedly frivolous, but possessing a clearly considered stash of great content, each issue hosts short interviews with compelling questions and illustrations from some of the web's better-known creatives.

The site was created by Brad Simon, a senior designer at Wieden+Kennedy, and Alexander Barrett, creative lead at YouTube. Simon explains he hopes visitors will "show up to their laptops with a plate of hummus and baby carrots, and experience the entire issue by the time their plate is empty."

"Early on we had a contributor write a longform piece about brunch," recalls Barrett, when asked about their biggest challenge. "Brunch is definitely not a snack. We felt a little funny telling her to make her piece snackier, but we did. It's all about retaining snack integrity."

The site would benefit from a responsive execution (the lack of which will be punished by Google's new-ish algorithm), but do organic search results really matter when content is so magnetic?

Matt Stevens, who illustrated a gallon of cheese for the site, participated because "it provided an opportunity to try something a bit goofy". Stevens demonstrates the sentiment, core to this site, that the web doesn't have to take itself seriously all the time.

WORDS: Cassie McDaniel

"[The web] is not all banks and law offices. You need the weird gift shop too"
MATT STEVENS (@MATTSTEVENSCLT)

LONDON 17-18 SEPTEMBER 2015

Explore CSS, UX, web performance strategies, the Internet of Things, app icon design and much more!

ERIC MEYER
CONSULTANT AND CSS PIONEER
meyerweb.com

RACHEL ANDREW
CO-FOUNDER, EDGEOFMYSEAT.COM
rachelandrew.co.uk

SARA SOUEIDAN
FRONTEND DEVELOPER, WRITER AND SPEAKER
sarasoueidan.com

TICKETS ON SALE NOW!

Follow @netmag for details

generateconf.com/london-2015

net

MAGAZINE IPAD EDITION

The iPad edition of **net** has been completely rebuilt from the ground up as a tablet-optimised reading experience.

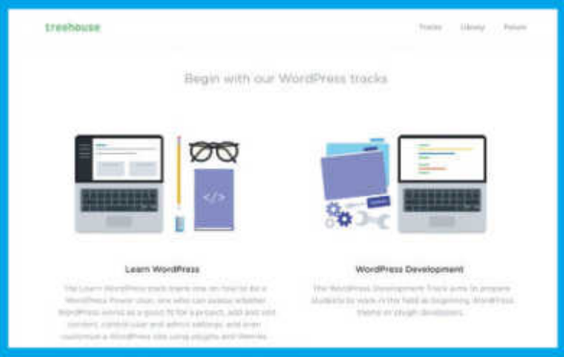
TRY IT FOR FREE TODAY!

You'll find additional imagery, exclusive audio and video content in every issue, including some superb screencasts that tie in with the practical projects from the issue's authors. **Don't miss it!**

TRY IT FOR FREE TODAY WITH OUR NO-OBLIGATION 30-DAY TRIAL AT
UK: netm.ag/itunesuk-270 US: netm.ag/itunesus-270

Available on the App Store

GETTING STARTED



Best sites

BUILD THE BEST WORDPRESS SITES

Shannon Smith presents four ways to improve your WordPress site using the latest development techniques

AUTHOR
SHANNON SMITH
Shannon is the founder of Café Noir Design, a boutique web design company that specialises in multilingual web development
shannon-smith.ca

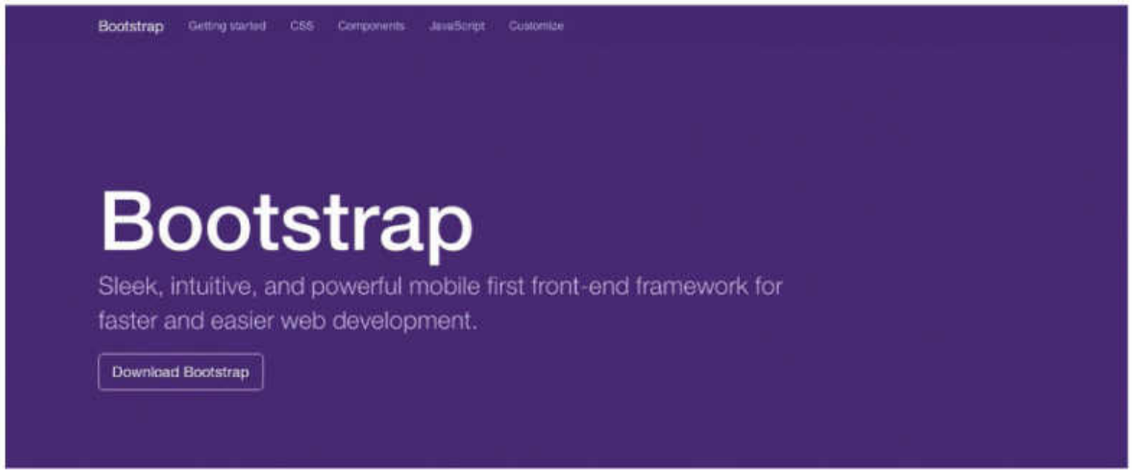
WordPress is the most commonly used online publishing platform on the planet. Millions of people view billions of WordPress pages every month. It used to be that many of those pages were on smaller blogs, but WordPress is becoming a more and more sophisticated tool. And with more complex demands, developers are bringing some of the most up-to-date development techniques to WP sites. We'll look at four of them.

Right Bootstrap is an open-source framework designed for fast and consistent website development

Below left The Schema Creator plugin allows you to add microdata to your posts and pages quickly

Below right Genericon'd allows you to use the Genericons icon font set from within WordPress

Far right Genericons contains 94 embedded vector icons



1 USE STRUCTURED DATA

Tim Berners-Lee once said, "I have a dream for the web [in which computers] become capable of analysing all the data on the web." Structured data, including microdata, is one way of providing the context that machines, including search engines, need to analyse all the web content that we produce.

Microdata is the newest form of structured data in wide use. Introduced with HTML5, microdata is essentially a set of metatags, which provide context to search engines.

Google has been promoting structured data as an effective SEO strategy and built a Structured Data Testing Tool (*netm.ag/rich-271*). You can find ways to tag various different types of data, including addresses, products, places and events on *Schema.org*.

One of the easiest ways to add structured data to your websites is with the Yoast SEO plugin (*yoast.com/wordpress/seo*). This enables you to add Google authorship data, Twitter Cards (*dev.twitter.com/docs/cards*) and Facebook Open Graph (*netm.ag/graph-271*) data just by filling in a few fields. Other plugins exist for more specific functions, like adding structured data to event and real estate listings, or to recipes.

Another good plugin is the Schema Creator plugin (*wordpress.org/plugins/schema-creator*). It allows you to insert *Schema.org* microdata directly into WordPress pages and posts.

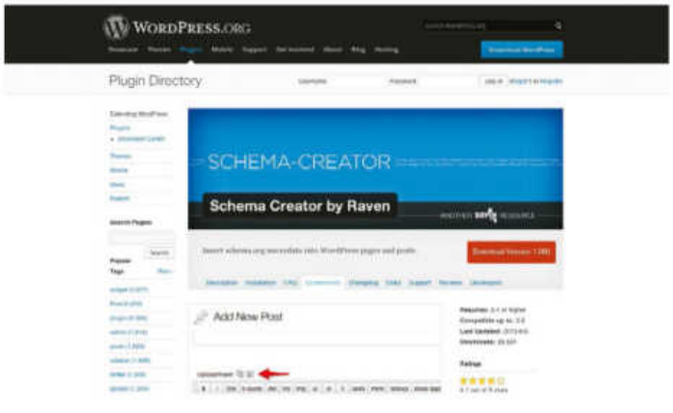
Structured data and themes

All WordPress sites have some structured data that comes from the core installation – for example, in RSS feeds. A few frameworks like Roots (*roots.io*) include microformats in the template files.

However, often if you want to add structured data to your site, you will need to add the necessary code to your template files. One of the best places to do this is in post type template files in combination with custom meta boxes.

2 USE ICON FONTS

The folks at Automattic released Genericons (*genericons.com*) as part of the Twenty Thirteen



theme. Genericons are vector icons embedded in a web font. They're free and, because they're licensed under the General Public Licence (GPL), you can use them in commercial projects.

The set comes with Sass and Less syntax examples. The download includes an OTF version, but run the set through the Webfont Generator at Font Squirrel (*netm.ag/squirrel-271*) for better cross-device compatibility.

Plugins exist for more specific functions, like adding structured data to event and real estate listings, or to recipes

Open-source icon fonts and licensing

Most open-source fonts are licensed under the SIL Open Font Licence, which is GPL-compatible. Font Awesome (*netm.ag/awesome-271*), the icon font bundled with the Twitter Bootstrap framework, is one. Iconic (*github.com/somerandomdude/Iconic*) and Entypo (*entypo.com*) are others. There are more available at Font Squirrel (*fontsquirrel.com*). However, some open source fonts are also released under other licences.

If you're building a custom theme for a client, the licence probably doesn't matter, but if you want to submit a theme to the official WordPress Theme Directory (*wordpress.org/themes*), or sell it commercially, you need to stay clear of fonts that are not GPL-compatible. Find a partial list of compatible licences and icon fonts on the *WordPress.org* website (*netm.ag/list-271*).

THEMES AND FRAMEWORKS

PORTED FRAMEWORKS AND THEMES

WP-Foundation

The ZURB Foundation framework was developed for prototyping responsive designs across a variety of devices. WP-Foundation (*320press.com/themes/wp-foundation*) is Foundation in a WordPress theme: it's a mobile-first design tool that incorporates a flexible grid and the lighter JavaScript library Zepto (*zeptojs.com*) instead of jQuery. It's also Sass-ready.

Underscores

Brought to you by Automattic (*automattic.com*), Underscores (*underscores.me*) is an ultra-minimal starter theme designed to give you a thousand-hour headstart developing your next site. This website even allows you to customise your files before you download them.

Roots

The Roots theme (*roots.io*) uses Bootstrap, HTML5 Boilerplate, ARIA roles and microformats all in a single framework that works with Grunt and Less. It also minifies and concatenates CSS and JavaScript for you.

Bones

Bones (*themble.com/bones*) is a theme for developers built around the HTML5 Boilerplate. Mobile-first and responsive, it comes loaded with Less, Sass, custom post types and custom dashboard functions.

Skeleton

Based on the minimalist Skeleton framework, this theme (*themes.simplethemes.com/skeleton*) aims to help you build simple, uncluttered, useable WordPress sites, that are also mobile-friendly.

Boilerstrap

Boilerstrap (*getboilerstrap.com*) is an open-source WordPress template based on Twenty Twelve that comes with Bootstrap, Font Awesome icons, and more.

UNPORTED FRAMEWORKS

HTML KickStart

This framework (*99lime.com/elements*), which bills itself as 'ultra–Lean HTML5, CSS and JavaScript building blocks for rapid website production' hasn't yet been ported to a WordPress theme, but it's so lean that it could easily be integrated into a number of starter themes.

Gravity

A Sass-based framework (*gravityframework.com*) designed to build powerful and easy-to-maintain HTML5 websites. It's designed for rapid prototyping, but hasn't yet been ported to WordPress.

ACCESSIBLE WORDPRESS SITES

Is your WordPress site accessible? There are several reasons to make sure that it is. The more people that can access your great content, the better. There are legal requirements for some organisations and government websites. An accessible site can be great for search engine optimisation, and some accessibility measures just make websites easier to use for everyone. Building accessible WordPress sites doesn't have to be difficult, but many people aren't sure where to begin.

Accessibility team

The WordPress project has an accessibility team (*make.wordpress.org/accessibility*) which is working to make the WordPress core more accessible by suggesting improvements to the WordPress UI. The team has compiled a list of resources and testing tools (*netm.ag/testingtools-271*) as well as planning coding and styling guidelines for accessible sites. It has also established a formal outreach effort for developers.

Official theme accessibility audit

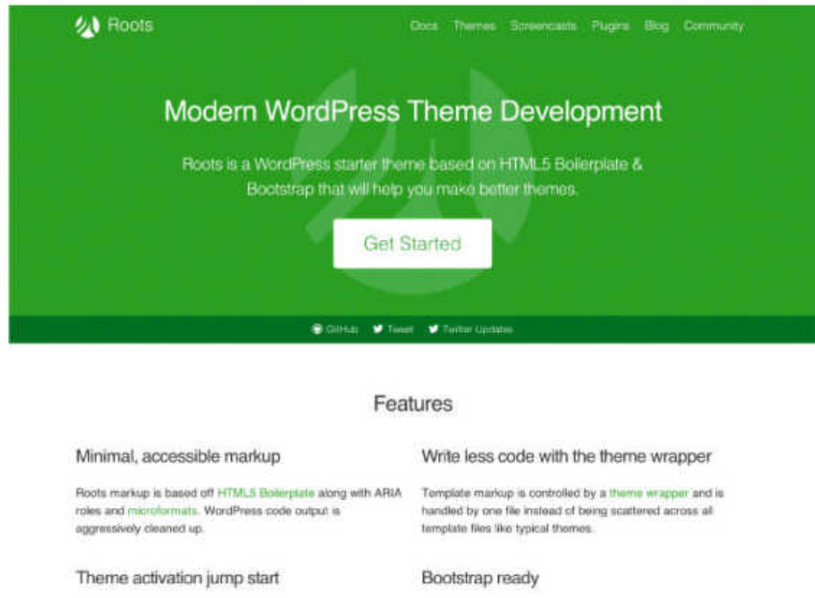
The WordPress theme review team tests and approves the themes that make it into the official directory. At the moment, if you search for 'accessibility' in the directory, 15 themes come up in the search results. This number has steadily been growing over recent years, and continues to do so.

The group has published a set of guidelines (*netm.ag/guideline-271*) that any developer should be able to incorporate into a theme with minimal effort. For example, developers are advised to include informative alt text, to prevent repetition of link text (such as the default 'Read more' links) and to check colour contrast. The idea is to make themes easier to use for people with visual impairments as well as for those using screen readers and keyboard navigation.

The WordPress codex has additional information (*codex.wordpress.org/Accessibility*) along with some examples. The review guidelines are an optional step in the official theme accessibility audit for themes submitted to the *WordPress.org* theme repository. It's probably best to think of this as a standardised process to ensure your theme has bare-bones accessibility.Themes submitted to the directory that pass the audit will get an 'accessibility-ready' tag that will help people find those themes.

Accessibility plugins

The main plugin used to increase accessibility is the WP Accessibility plugin. The plugin adds common accessibility features to most themes and corrects the most common accessibility issues at the same time. For example, it adds the post titles to 'more' links to make them more useful for people using screen readers. There are also a number of other accessibility plugins (*wordpress.org/plugins/tags/accessibility*) in the official WordPress repository.



Icon font plugins and services

Is there a plugin? Yes! Genericon'd (*wordpress.org/plugins/genericond*) enables you to use the Genericons font with shortcodes. There are also a number of plugins that will let you use Font Awesome with shortcodes. If you only want a few icons, Fontello (*fontello.com*) is an online service that will generate a custom bundle of icons from some of the most commonly used icon fonts.

Of course, if you can't find what you need for your particular project, you can always generate your own icon font. Inkscape (*inkscape.org*) is an open source vector graphics tool, and it has a SVG font editor that will enable you to you turn vector graphics into a font for free.

There's even an icon font starter template available (*github.com/Heydon/Community-Icon-Font*). There are also a number of online tools, like IcoMoon (*icomoon.io*), to make the whole process easier.

3 USE BOOTSTRAP WITH WORDPRESS

The Bootstrap framework (*getbootstrap.com*) bills itself as a 'sleek, intuitive and powerful frontend framework for faster and easier web development'. It includes UI elements and several responsive layouts. It utilises Less, CSS and icon fonts, and includes a responsive grid. It's also a favourite with startups looking for a quick way to prototype. So why add WordPress? I'll tell you why: to make the process even faster.

If you just want the CSS and JavaScript libraries, you can easily use the WordPress Twitter Bootstrap CSS plugin with an existing theme (*netm.ag/*

```
 * @package WordPress
 * @subpackage Twenty_Twelve
 * @since Twenty Twelve 1.0
 */

get_header(); ?>

	<div id="primary" class="site-content">
		<div id="content" role="main">

			<article id="post-0" class="post error404 no-results not-found">
				<header class="entry-header">
					<h1 class="entry-title"><?php _e( 'This is somewhat embarrassing, isn&rsquo;t
					it?', 'twentytwelve' ); ?></h1>
				</header>

				<div class="entry-content">
					<p><?php _e( 'It seems we can&rsquo;t find what you&rsquo;re looking for.
					Perhaps searching can help.', 'twentytwelve' ); ?></p>
					<?php get_search_form(); ?>
				</div><!-- .entry-content -->
			</article><!-- #post-0 -->

		</div><!-- #content -->
	</div><!-- #primary -->

<?php get_footer(); ?>
```

Opposite Starter theme Roots is based on HTML5 Boilerplate

Above Plain text strings from the 404 error page are wrapped in WordPress-defined PHP functions

exist-271). There are a number of Bootstrap themes that have been developed for WordPress. WordPress Bootstrap (*netm.ag/wpbootstrap-271*) includes Bootstrap as well as the additional layouts available as Bootswatch themes (*bootswatch.com*). It also includes shortcodes, page templates and sidebars.

BootstrapWP (*bootstrapwp.rachelbaker.me*) is another option. Prefer to use Sass rather than Less? You can find a WordPress theme for that too: it's called Sass WordPress Bootstrap (*netm.ag/bass-271*).

Roots (*roots.io*) is another WordPress Bootstrap theme, although it's designed as a minimally styled starter theme that includes Bootstrap, HTML5 Boilerplate, ARIA roles and microformats.

It also works with a number of preprocessors including Less, Less with pure CSS, Sass and Compass. Roots works with Gravity Forms and WooCommerce, the leading form and ecommerce plugins for WordPress. It has already been internationalised for at least 23 languages, too.

> WordPress uses the open-source GNU gettext framework to prepare special files for translation purposes

4 CREATE MULTILINGUAL SITES

Why only speak to a few people when you can speak to the whole world? Most of the world population uses the internet in a language other than English. WordPress itself is already available in 151 languages (*netm.ag/languages-271*).

To get started, all you need to do is download the language files for the language you want and add them to your WordPress installation. If the language you require isn't available, there are teams working on supporting even more, and community participation is encouraged.

Internationalise WP themes and plugins

The default themes, like Twenty Twelve and Twenty Thirteen have already been translated into a number of languages that makes them great starter themes. However, if you want to develop a new theme that's ready for the world, internationalising WordPress themes isn't especially difficult. Internationalisation (I18n) is simply the process of making an app, or in this case a theme, ready for translation.

WordPress uses the open-source GNU gettext framework to prepare special files for translation purposes. As a theme developer, you need to do three things. First, assign a text domain to your theme and load it in your 'function.php' file:

```
<?php load_theme_
textdomain( $domain, $path ) ?>
```

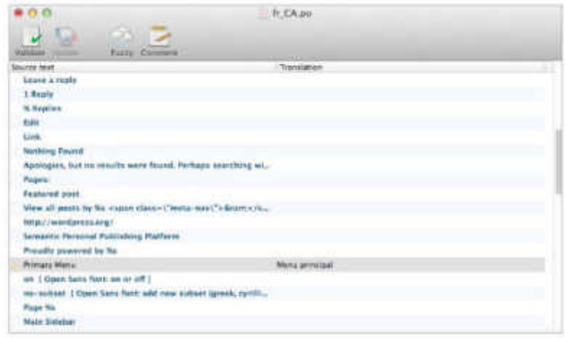
WordPress-defined PHP functions

Next, wrap any text in your theme in WordPress-defined PHP functions. There are a couple of

different ways to wrap the text depending on whether you are wrapping plain text (use `visloop _e($text_message)` in that case), or text that will be passed to another function (use `__($text_message)` for that text).

There is some information on which to use in the WordPress Codex (*netm.ag/wpphp-271*), but this is the basic syntax:

```
<?php $translated_text = __( 'text_
message', 'domain' ); ?>
```



It's also possible to generate phrases that use placeholders, and which use different values for singular and plural words, and to add context information for translators.

Finally, you need to generate a POT (Portable Object Template) file for your theme. This file contains a list of all the bits of text in a theme or plugin that need to be translated and includes every message passed into `a __() or _e()` function.

This file can be generated in several ways, but the most common is to use the open-source application Poedit (*poedit.net*). There is also a plugin called the Codestyling Localization plugin (*netm.ag/localization-271*) that will enable you to create this file very easily.

That's it! The same general rules apply to internationalising plugins, though these can be quite complex and may require more careful wrapping of text.

Translate themes and plugins

Localisation (L10n) includes all the steps needed to actually translate a theme or plugin. As long as a POT file has already been created, a translator can use a tool like Poedit to go in and translate each string of text in a theme. As long as the text in is phrases, translators can easily account for changes in word order and other linguistic particularities.

The translated text is saved in new, language-specific PO (Portable Object) file. It's also good practice to generate an MO (Machine Object) file for each language. This is simply a copy of a PO file written in binary, and it allows for faster processing.

GETTING STARTED WITH PREPROCESSORS AND WORDPRESS

CSS preprocessors are tools designed to make CSS authoring more efficient. They enable you to speed up your development by using nested rules with CSS selectors and mixins. Mixins allow you to treat CSS selectors as variables, where properties can be extended to other selectors.

Using preprocessors means fewer lines of code and less repetition. Preprocessors compile the code you write into a standard CSS stylesheet, which can also be minified at the same time.

There are several preprocessor options available. Less (*lesscss.org*) and Sass (*sass-lang.com*) are the most widely used. There's also Stylus (*learnboost.github.io/stylus*). Compass (*compass-style.org*) and Sass syntax SCSS are also useful.

Which one should I use?

If you work in a development team, your team has probably already decided for you. If not, you should probably try Less and Sass. In any case, the syntax for the two is quite similar. Compass is a framework for Sass that makes it easier to use.

Once you have chosen a preprocessor and have the syntax down, it's time to find an app so that you can start creating some code. Both Less and Sass can be used without a nice graphical interface, but why not make it easy for yourself?

The easiest interface

Both Less and Sass can be used in the CSS editor on *WordPress.com*, which comes with the Jetpack plugin (*wordpress.org/plugins/jetpack*) for self-hosted blogs. Just change the Preprocessor option when editing your stylesheets within the WP dashboard.

Choosing an app

You can also use any number of online tools to use as a preprocessor, but you will probably prefer to have a dedicated app. Compass (*compass.handlino.com*) is a menu bar-only app for Sass and Compass that works across various platforms.

Prepros (*alphapixels.com/prepros*) is an app that works with many preprocessor file types. The most widely used apps seem to be CodeKit (*incident57.com/codekit*) for Macs and LiveReload (*livereload.com*) for Windows.

More information

Here are some further useful links:

- 'Getting started with Sass' (*alistapart.com/article/getting-started-with-sass*)
- 'Using the Less CSS preprocessor for smarter style sheets' (*netm.ag/less-271*)
- 'How to: Get started with the Compass CSS framework' (*mashable.com/2011/06/14/compass-css-guide*)
- 'Setting up CodeKit for Sass' (*netm.ag/unmatched-247*)

```
functions.php
32  * Sets up theme defaults and registers the various WordPress features that
33  * Twenty Twelve supports.
34  *
35  * @uses load_theme_textdomain() For translation/localization support.
36  * @uses add_editor_style() To add a Visual Editor stylesheet.
37  * @uses add_theme_support() To add support for post thumbnails, automatic feed links,
38  *  custom background, and post formats.
39  * @uses register_nav_menu() To add support for navigation menus.
40  * @uses set_post_thumbnail_size() To set a custom post thumbnail size.
41  *
42  * @since Twenty Twelve 1.0
43  */
44 function twentytwelve_setup() {
45     /*
46      * Makes Twenty Twelve available for translation.
47      *
48      * Translations can be added to the /languages/ directory.
49      * If you're building a theme based on Twenty Twelve, use a find and replace
50      * to change 'twentytwelve' to the name of your theme in all the template files.
51      */
52     load_theme_textdomain( 'twentytwelve', get_template_directory() . '/languages' );
53
54     // This theme styles the visual editor with editor-style.css to match the theme style.
55     add_editor_style();
56
57     // Adds RSS feed links to <head> for posts and comments.
58     add_theme_support( 'automatic-feed-links' );
59
```

Far left As with all new versions, WordPress 4.0 improves upon the past by fixing hundreds of minor user-reported issues

Middle The Customizer's new Widgets panel allows for quick and easy management of sidebars and widgets – complete with live previews

Left The rebranded and redesigned Customizer allows theme developers to control every aspect of the frontend of their site

Above left A translator can view a POT file and translate each string of text using software like Poedit

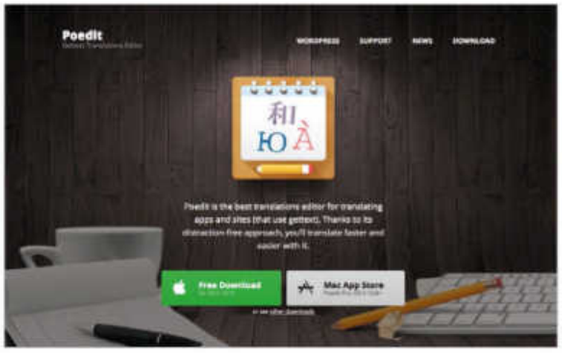
Above The Twenty Twelve WordPress theme uses the text domain 'twentytwelve', declared in the 'functions.php' file

Below left Translations editor Poedit uses gettext for localisation

GlotPress (*glotpress.org*) is a web tool based on WordPress that's aims to facilitate the translation process. It's installed online (or locally), just like WordPress, and is currently being used to help translate parts of *WordPress.com*.

The WordPress Multilingual Plugin allows you to run a site that operates in many languages at once

It's intended for collaborative translation, but single translators can use it as well. It helps users translate POT files and create PO and MO files, and it's available for download at *glotpress.trac.wordpress.org*.



Unilingual WordPress sites

Having a unilingual site in a language other than English is a great start, but many people want to offer content in multiple languages at once. There are currently two preferred ways of doing this.

The first option is to use the WordPress Multilingual Plugin (*wpml.org*), which is a commercial plugin that allows you to run a site that operates in many languages at once, including the theme and plugins as well as basic core functionality. It works with most complex plugins including Gravity Forms (*gravityforms.com*) and various ecommerce plugins.

The other option is to use the multisite functionality built into WordPress to run multiple WordPress installations for each language, all using the same theme and plugins. This option is faster and more stable, but may not have all the functions you need.

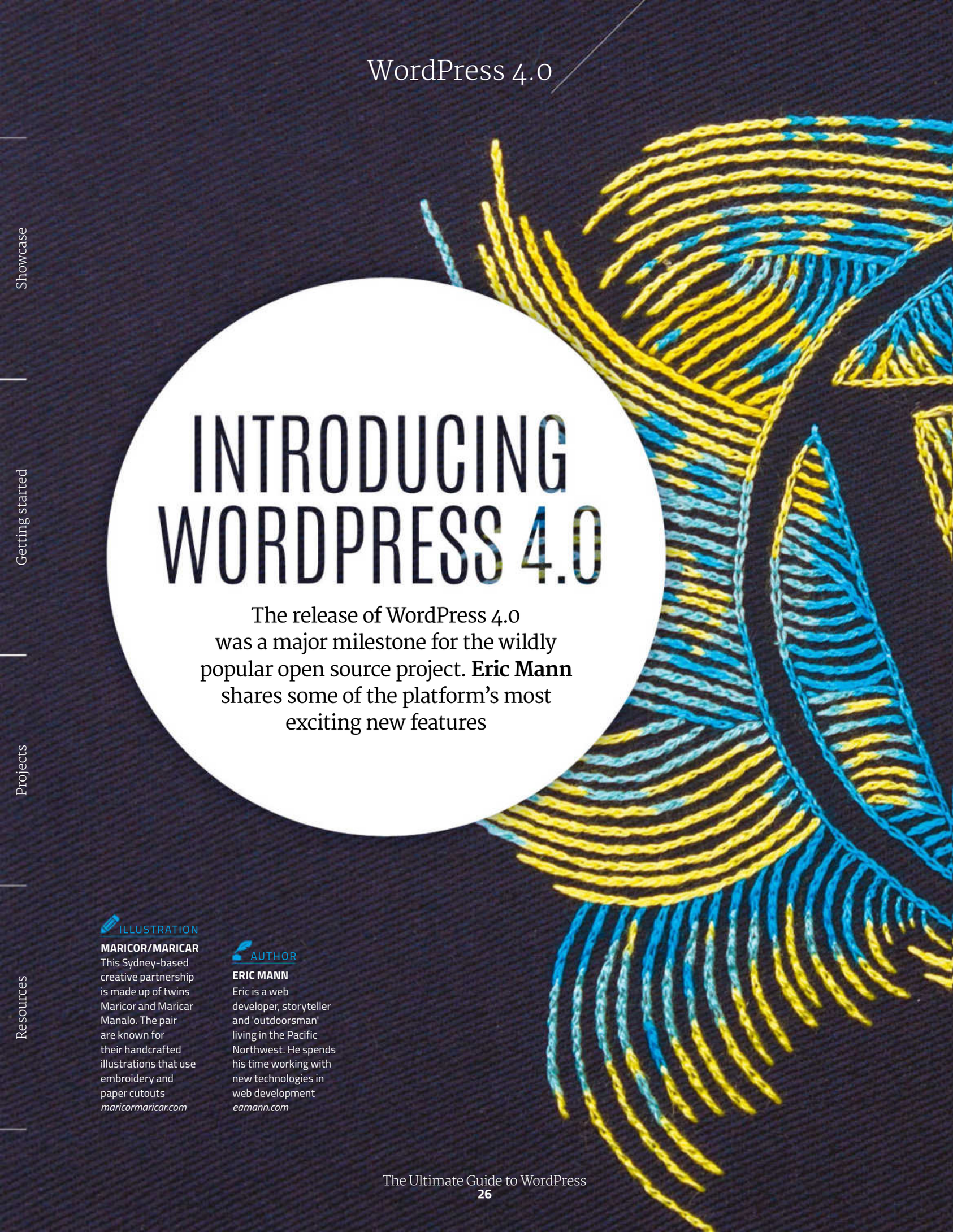
There are several other plugins that could be used to build a multilingual site. However, they tend to be unsupported, unproven, or lacking in the functionality compared to the two solutions described here.

RESULT: A FUTURE-PROOFED SITE

From structured data to open-source icon fonts and Bootstrap, hopefully this article has introduced you to a few valuable new methods and techniques you can apply to your development to help you create better WordPress websites, that will set you in good stead for the future.

INTRODUCING WORDPRESS 4.0

The release of WordPress 4.0 was a major milestone for the wildly popular open source project. **Eric Mann** shares some of the platform's most exciting new features



ILLUSTRATION

MARICOR/MARICAR
This Sydney-based creative partnership is made up of twins Maricor and Maricar Manalo. The pair are known for their handcrafted illustrations that use embroidery and paper cutouts
maricormaricar.com

AUTHOR

ERIC MANN
Eric is a web developer, storyteller and 'outdoorsman' living in the Pacific Northwest. He spends his time working with new technologies in web development
eamann.com

WordPress 4.0

Back in 2007 I started writing a blog on WordPress. I've been working professionally with the platform, and contributing code, since 2010. In this time, I've seen 17 major releases come and go. Still, nothing excited me more, as a writer and as a developer, than the new feature set for WordPress 4.0.

The core team announced the start of WordPress 4.0's development at the end of April 2014. The 23rd release in its history, WordPress 4.0 was heralded as a landmark '.0' release. It was led by a talented, UX-focused engineer and backed by hundreds of savvy developers across the globe. The initial slate of feature proposals was enough to excite every WordPress fan, and watching the update being narrowed down to a set of well-coded, well-tested, well-vetted features was incredible.

As with any new release, there are features all users love and those we're less than excited about. For the first time I can remember, every one of version 4.0's features is something to talk about. I have, though, whittled down my list of "I can't believe we finally have this" to what I believe are the most important new features.

1 INSTALL IN YOUR LANGUAGE

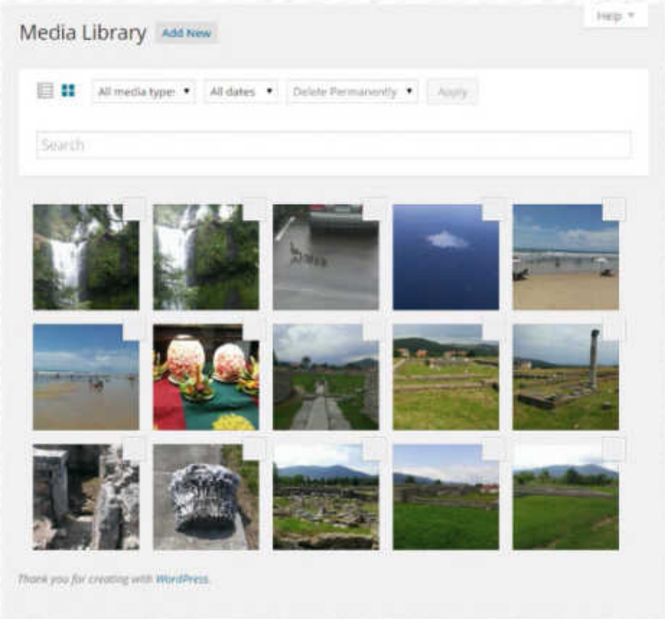
One of the overarching goals of the WordPress platform is to "democratise publishing". Features ship in every version to make it easier for writers to put their voice online under their own terms. As the majority of the world communicates in languages other than English, this means the WordPress admin needs to present content in a non-English format.

There is a large contingent of volunteers that works tirelessly to translate text in WordPress to various languages. Many themes and plugins also tie in to WordPress' translation features, making it possible for non-English speakers to use the entire platform, including optional extensions.

Up until 4.0, though, the installation and setup process for WordPress was English-only. New users either had to use English throughout the installation, or find someone to walk them through the process. As you can imagine, this was a major barrier to entry for non-native English speakers.

> Nothing excited me more than the new feature set for WordPress 4.0

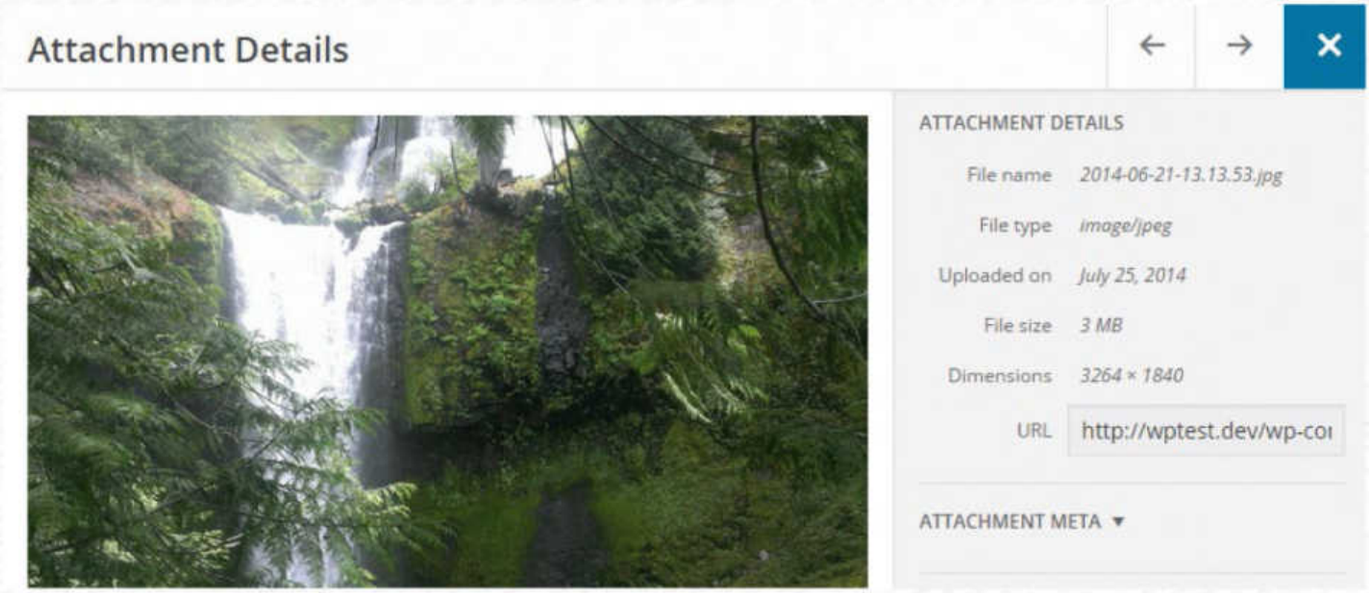
WordPress 4.0 changed this. During initial setup and installation, the platform now allows users to specify their primary language. Selecting a language other than English triggers the download of a language pack, automatically converting instructions throughout the rest of the installation screens – and the WordPress admin itself. The core WordPress community recognises the value of our growing international audience. For the first time, non-native English speakers are able to install and configure WordPress without needing to hire an expensive contractor to set things up for them.



Below left The first screen presented by WordPress 4.0's installer allows for language selection, automatically translating the rest of the installer

Below right The new Media Library grid view includes larger thumbnails, making it easier to manage assets in the admin

Far right The Media Library's grid view displays media details in a convenient modal window



These changes have made WordPress more accessible to more users than ever before. As May of this year marked the first time non-English downloads of the platform exceeded English downloads, we can only expect further international adoption for WordPress from this point forward.

2 MEDIA MANAGEMENT

WordPress 3.5 introduced a completely reimagined media gallery that made it far easier to both import and work with images inside of posts and in the admin itself. WordPress continues to optimise the media experience in version 4.0 by introducing a new media grid to the gallery.

In the past, editors only had one option for viewing items in the Media Library – the list view. This presented a paginated list of items, each with relatively small thumbnails. Managing media on a large site meant navigating a somewhat boring list of images spanning several pages. If you've been using WordPress for a while, you'll know the pre-4.0 media list view was a less than ideal experience.

The updated Media Library enables the new grid view by default. This view presents a rich grid of larger image thumbnails that pulls in subsequent pages of content using the same approaches we employ for infinite scroll. Clicking any particular thumbnail loads the attachment's meta information in a modal window. This further empowers rapid editorial changes without the need to navigate within WordPress to different pages.

Without this meta information modal, managing individual assets in bulk was tedious and time-consuming. Even viewing asset captions and

INSIDER'S VIEW: Helen Hou-Sandi

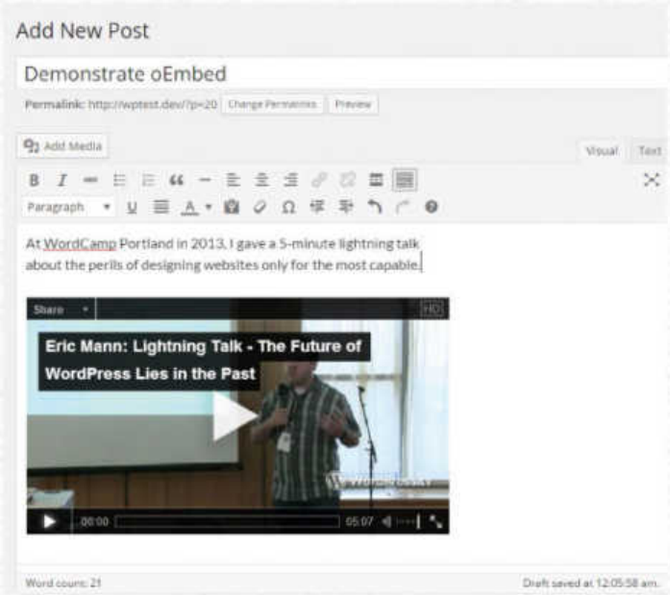
After the second beta release, I had the opportunity to sit down with Helen Hou-Sandi, the release lead for WordPress 4.0, to learn a little how she'd seen the project evolve over the past few months. Past versions of WordPress often had themes for the release to guide the overall development of the project. Version 4.0, however, had less of a central theme.

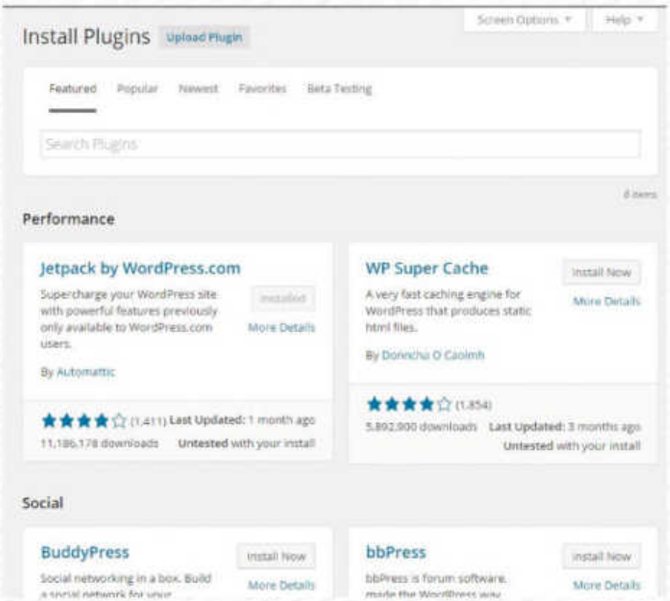
"We've been better at iterating on what we've already got," Hou-Sandi explains. As a result, most features scheduled for the release build on past successes, or focus on refreshing elements.

An example of the latter is WordPress' plugin interface. According to Hou-Sandi, plugins are "central to what WordPress can do", but the interface has remained relatively untouched since 2009. She explains that one of the biggest focuses of 4.0 has been to improve the experience for new users, and that revamping the way plugins are both discovered and managed within WordPress has been high on the development team's list.

Ultimately, Hou-Sandi says, the objective of 4.0 has been to "do small things continually on all fronts" rather than focusing exclusively on any one area. She hopes the final product will benefit multiple groups – making WordPress simpler for newer users while also extending powerful new APIs for developers.

WordPress is very much a living project, making continued progress on various fronts with each new release. The goal for 4.0 is the same as every version: to make WordPress better.





HOW TO GET INVOLVED

WordPress has a gruelling release cycle, targeting a new version every four months. Getting stellar new features into WordPress core on such a limited timetable takes a lot of work, and requires a lot of community involvement.

Want to get involved in the development of code for WordPress? The best way to get started is by following the core development blogs:

- WordPress Core (*make.wordpress.org/core*)
- WordPress UI Group (*make.wordpress.org/ui*)
- WordPress Accessibility Group (*make.wordpress.org/accessibility*)

Even non-developers can contribute to ongoing development. Features need to be tested, bugs need to be reported and systems need to be documented. You can help build the next version of WordPress, even if you never write a single line of code.

Want to help keep WordPress rolling? Start by contributing in some of the following places:

- WordPress Support Forum (*wordpress.org/support*)
- WordPress Code Reference (*developer.wordpress.org/reference*)
- Get involved with your local Meetup group (*wordpress.meetup.com*)
- Attend or volunteer at a WordCamp near you (*central.wordcamp.org*)

descriptions forced a separate page load. The refined experience presents all attachment details in an easy-to-access location that saves time and effort.

Even non-image attachments like documents and Zip archives benefit from the new grid treatment. While they lack individual thumbnails, presenting MIME-type icons in the same format as the larger image thumbnails definitely makes the Media Library as a whole feel more unified.

3 EDITOR UPGRADES

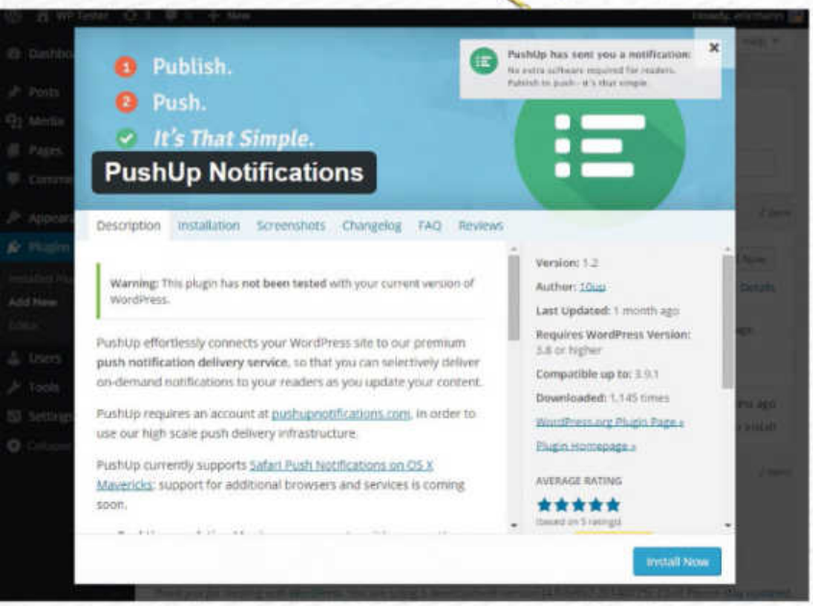
The most-used feature of WordPress by far is the WYSIWYG content editor. Whether you use WordPress for casual blogging, professional news media, or merely to curate a static body of content, you'll at some point have used the editor itself.

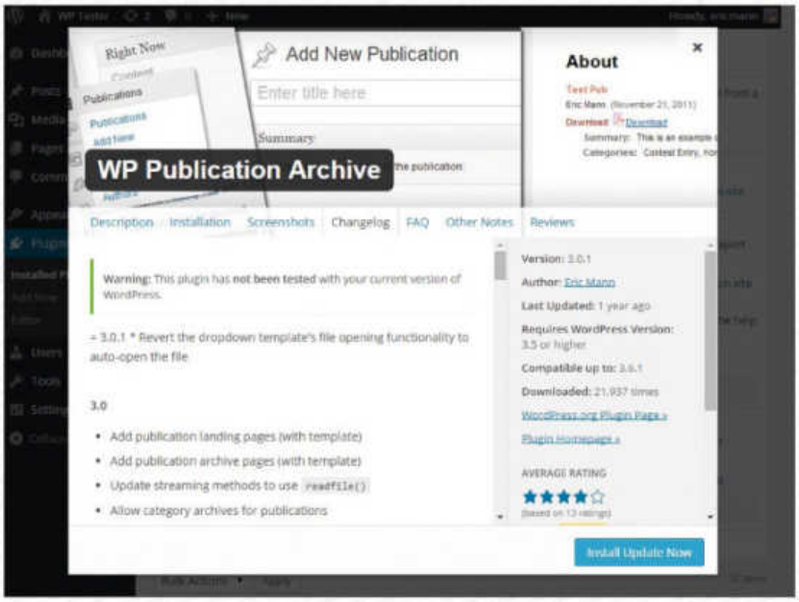
As the most recognisable element of the platform, you'd expect the editor to receive a fair amount of polish. WordPress 4.0 brings several updates to the editorial experience and the editor itself. There are two exciting changes to keep an eye on.

oEmbed previews

WordPress has long been a champion of using oEmbed to easily share content from one site or platform to another. The 4.0 release iterated on existing support to add new oEmbed providers for sites like *ted.com* and *collegehumor.com*. It also drastically evolved the way we experience oEmbed within the content editor.

In the past, editors could merely copy and paste the URL of an oEmbed-read resource (like a YouTube video or Twitter message) directly into the post editor. When viewed on the frontend, this raw





Opposite left When oEmbed-capable links are present within post content, WordPress will generate a preview of the linked resource

Opposite right The new plugin installation page presents more streamlined discovery tools – starting with better categorisation of available plugins

Above left The details modals for plugins display richer information than in previously – including the plugins' graphical banners

Above right When plugins are ready for upgrading, a More Details link exposes the plugin's changelog, ratings and compatibility information

URL would be dynamically converted to a richer representation of the linked material. The feature was a huge value-add for editors linking to external content, but also made it a bit tricky to optimise body content around an embedded resource.

WordPress 4.0 changed this feature up by enabling live previews of the embedded content. Once the link is pasted into the editor, WordPress automatically scans the link, fetches the associated resource through oEmbed, creates a representative preview and displays that preview in the editor.

> The top toolbar will pin itself to the top of the page even as your content flows on

It's an actual implementation of WYSIWYG for the WordPress editor that eliminates the pain of using oEmbed to extend post content. The original URL for the embedded asset will always be available on the Text view; but once you've seen the inline preview, you'll never want to leave the Visual editor.

Sticky toolbar

My favourite new feature in WordPress isn't so much an addition as a refinement of existing behaviour. In WordPress 4.0, the toolbars for the visual editor – both at the top and the bottom of the content editor – stick to the browser window when you scroll the page. As you write and your article grows longer, the content area automatically expands to contain every word. The top toolbar – with rich text formatting options and media embed features – pins itself to the top of the page as your content flows on. No more scrolling in the window to change paragraph formatting or insert links.

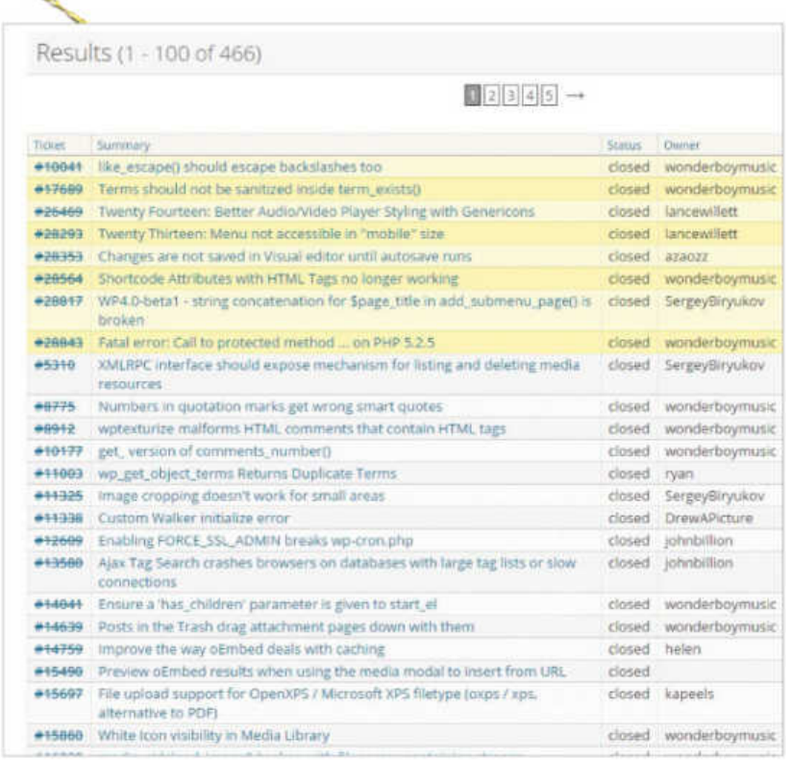
While scrolling back to the top of the article, the footer of the content editor pins itself to the bottom of the browser window. Whether you're editing content in the beginning of the post, the end of the post or somewhere in-between, you retain access to both your formatting tools and after-the-post meta information like word counts.

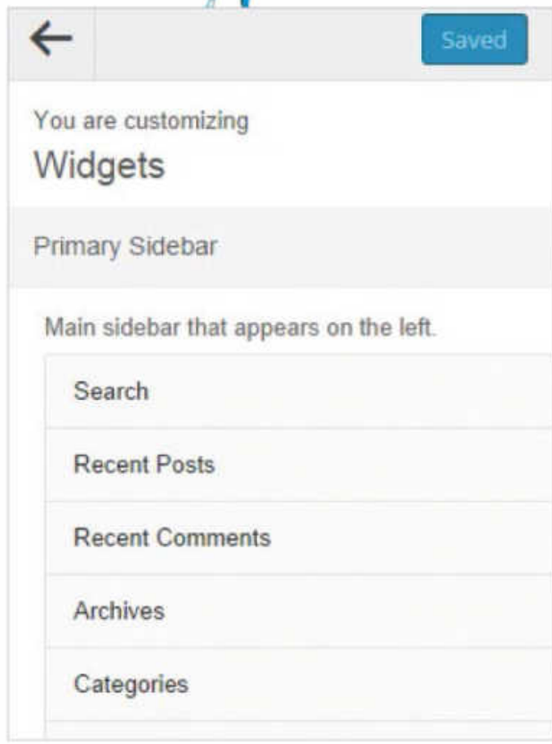
This is a somewhat minor change in the editorial experience that has major benefits for the writers of long-form content. Once you use the new 'sticky' editor, you'll question why it took this long for the feature to evolve.

4 ENHANCED PLUGIN EXPERIENCE

I started my career with WordPress as a plugin developer, so the end-user plugin experience has always been a key point of interest for me. Plugins are central to making WordPress function as a customisable platform, and as a contractor I spent a large number of hours teaching clients how to work with plugins. Unfortunately, the plugin screens in WordPress have not been the most user-friendly in the past. Worse still, they haven't received much more than a new paint job in the past five years.

WordPress 4.0 introduced a number of changes to the way new users experience plugins, both in terms of discovery and management. The plugin installer used to attempt to aid feature discovery





with a basicsearch field and essentially meaningless tag cloud. Unless you already knew what to look for, the chances of finding something new for your site were limited. The new installer highlights 'featured', 'popular' and 'new' plugins, with the featured plugins further broken down into categories like 'performance' and 'social'.

On the installation pages, plugins are now displayed in detailed tiles presenting their title, a brief description and the plugin author, as before. These tiles also highlight the number of downloads and average plugin rating. A clever More Details link launches an overlay displaying the plugin's full information as pulled from *WordPress.org* – including the plugin's graphical banner.

This details screen is also presented when plugin updates are available, making the plugin's changelog, ratings and compatibility information readily accessible to administrators.

5 CUSTOMIZER IMPROVEMENTS

The Theme Customizer underwent a strong set of changes with 4.0, and has been rebranded as just the 'Customizer'. According to the WordPress 4.0 Customizer Improvements infomation (*netm.ag/improvements-259*):

"'Customize' could refer to anything. That's the point ... the Customizer can be used for anything, and we'd like to encourage more experimentation with different uses of the Customizer."

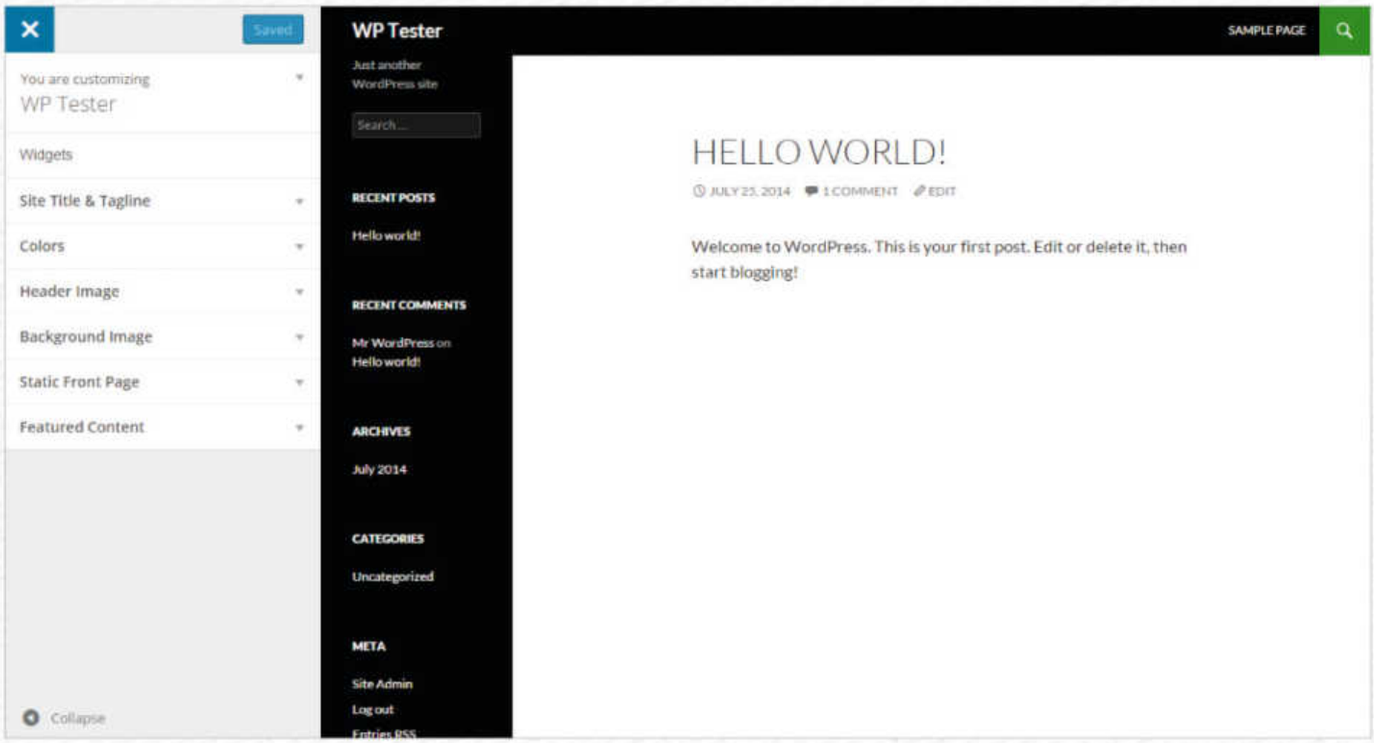
Widgets panel

Widget support was added to the Customizer in the previous version of WordPress, allowing site owners to see live previews of changes to sidebar widgets as they were made. Expanding the scope of the Customizer means we'll be adding more to the tool in coming releases, so widgets have earned their own Panel within the Customizer UI.

Panels are a way to group Customizer sections, allowing both cognitive and visual separation between different elements that are meant to be managed separately. Site administrators can dig into their theme to customise just the colours, banner imagery and content layout on one panel. Then they can delve deeper to manage the content and arrangements of widgets on another. Widgets are just the first of a promising series of changes to the Customizer system.

Developer API

While it seems like a minor refinement to the Customizer tool itself, the addition of a Panels API opens the door for developers to make future themes even more flexible. Every site owner wants a unique design, but not everyone has the time to build a custom theme – or the budget to hire a developer.



Far left As with all new versions, WordPress 4.0 improves upon the past by fixing hundreds of minor user-reported issues

Middle The Customizer's new Widgets panel allows for quick and easy management of sidebars and widgets – complete with live previews

Left The rebranded and redesigned Customizer allows theme developers to control every aspect of the frontend of their site

Flexible themes that support the Customizer are the easiest way for the majority of non-technical users to reach into their WordPress site and truly make it their own. Editors – and writers – can modify the design of their homepage, About page, content archives, and even more though the Customizer. The Customizer API adds visual finesse to the existing theme options API, empowering developers to truly enable customers to lift the hood on their site.

Being able to nest certain features under one or more Panels makes it easy to separate one type of customisation from others. For example, global theme features can live on one screen, contextual features that only present data to logged-in viewers can live on another.

The changes introduced by WordPress 4.0 mean the only limitation on what the Customizer can do is the imagination of the theme developer. Thankfully, theme designers tend to be on the creative side.

> Every new feature speaks to the power and ingenuity of the WordPress community

BOTTOM LINE

Most development communities place a lot of weight on '.0' releases. WordPress 2.0 saw a completely redesigned backend that introduced Ajax for a snappier administrative interface, and the WYSIWYG post editor we still use today. WordPress 3.0 saw the merger of the WordPress and WordPress MU (multi-user) code bases to present an improved platform capable of hosting multiple sites on one installation.

Some have argued that WordPress 4.0 isn't as significant a change as its '.0' predecessors. In truth, the core WordPress community never intended for any of its '.0' releases to be landslide events in the first place. Instead, WordPress focuses on small, rapid iterations of a large collection of features, releasing a new version every four months to maintain forward momentum on development.

This release happened to coincide with the number 4 – but that version number doesn't require any special requirements on the feature set. However, given the features outlined in this article, and the countless minor fixes and optimisations under the hood, WordPress 4.0 proved to be no smaller a milestone than any '.0' that's come before.

Every feature in this release speaks to the power and ingenuity of the community that maintains WordPress. These features also demonstrate just how much momentum WordPress will continue to carry into future versions.


CRAFT PERFECT THEMES

Corey Ellis walks through the dos and don'ts you need to know to plan, build and maintain a great WordPress theme

What is a WordPress theme? For anyone who has worked with another development framework, this question can be a mysterious proposition. With other frameworks – Drupal, Ruby on Rails, Symphony and so on – the separation of concerns principle is easy to discern. Whether it is the view in an MVC system or the presentation layer in an OSI model, the part of the software or framework that controls what the user sees is easily definable. While a WordPress theme does, in general, provide the presentation layer, it can do much more.

The fact that the bar to entry for WordPress is so low is both a gift and a curse. It is a gift because one can dive in and do amazing things without a high level of technical knowledge (I realise now that I built my first theme without any real understanding of PHP – I was a master of copy and paste!) It is a curse because as a platform it is so forgiving. You can go against the WordPress coding standards for PHP, HTML, CSS and JavaScript (*netm.ag/standards-267*) and it is possible that your theme will still work just fine.

In this article I will lay out some dos and don'ts to guide you to better development practices and a more thorough understanding of WordPress themes. I will explore how to structure your theme in a sane and straightforward way, and take a closer look at the WordPress templating system and how data is retrieved. Follow these rules and you'll be on your way to creating a perfect WordPress theme in no time.

DO SEPARATE FUNCTIONALITY AND PRESENTATION

Many WordPress theme authors try to cram all the functionality into their theme, when much of it belongs in plugins. I am very much an advocate of keeping a theme as close to the presentation/view layer as possible.

If the functionality I am working on does not provide visual enhancement or frontend display, it probably belongs in a plugin. Limiting theme functionality also enables users to more easily change the look of their site, without fear of losing a key piece of functionality.

What if you have a plugin that is tightly coupled with the theme? My approach here is to include styles for the plugin functionality in the theme. This approach can also potentially lower the number of http requests, which makes for better site performance.

There has been substantial debate around this topic, and if you want to explore further, take a look at Tom McFarlin's 'Should I do this in a WordPress theme or plugin?' (*netm.ag/mcfarlin-267*), Pippin Williamson's 'Functionality: Plugins vs themes' (*netm.ag/williamson-267*) or Sarah Goodling's

AUTHORS
COREY ELLIS
Corey is a senior frontend engineer at 10up. He's also the organiser of WordCamp Austin 2015 and helps run the WordPress Austin Meetup
coreyellis.me

ILLUSTRATION
ALEXANDRA BRUEL
Paris-based artist Alexandra is known for her work with modelling clay, and has worked with clients including *Vogue* and *Wired*
netm.ag/bruel-267

```php
namespace Core\ShadowTax\Show;

use Core\ShadowTax\TDS;

/**
 * Registers various hooks with the WordPress API
 *
 * @since 0.1.0
 *
 * @return void
 */
function load() {

    add_action( 'init', __NAMESPACE__ . '\register_taxonomy', 11 );

}

/**
 * Registers the Season Taxonomy for connections
 *
 * @since 0.1.0
 *
 * @return void
 */
function register_taxonomy() {

    $post_types = get_post_types();

    $show = array(
        'hierarchical'          => false,
        'label'                 => __( 'Show Shadow', 'core' ),
        'show_ui'               => false,
        'show_admin_column'     => false,
        'update_count_callback' => '_update_post_term_count',
```

```php
namespace Core\ShadowTax\Season;

use Core\ShadowTax\TDS;

/**
 * Registers various hooks with the WordPress API
 *
 * @since 0.1.0
 *
 * @return void
 */
function load() {

    add_action( 'init', __NAMESPACE__ . '\register_taxonomy', 11 );

}

/**
 * Registers the Season Taxonomy for connections
 *
 * @since 0.1.0
 *
 * @return void
 */
function register_taxonomy() {

    $post_types = get_post_types();

    $season = array(
        'hierarchical'          => false,
        'label'                 => __( 'Season Shadow', 'core' ),
        'show_ui'               => false,
        'show_admin_column'     => false,
        'update_count_callback' => '_update_post_term_count',
```

Avoiding conflict It is possible to have functions with the same name, as long as they exist in different namespaces, as seen here with load() and register_taxonomy()

'Why WordPress theme developers are moving functionality into plugins' (*netm.ag/goodling-267*).

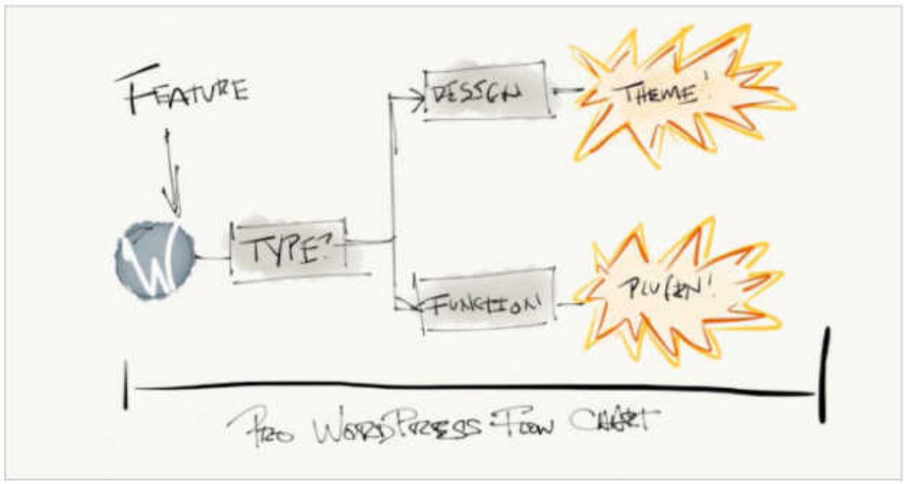
DO WRITE PROCEDURAL CODE

As I have become more involved in the WordPress community, I have seen a common thread in those of us who consider ourselves 'themers': many of us learned PHP through learning WordPress. WordPress is written in a mostly procedural manner, which means many of us understand procedural programming better than we do object oriented programming (OOP). But because WordPress still supports PHP 5.2 we tend to find ourselves either writing pseudo-OOP code or prefixing all our functions with something in order to prevent naming conflicts.

I think we should to try and push our community forward and not limit ourselves because of a version of PHP that has been unsupported for over four years (*netm.ag/version-267*). I encourage you to use PHP namespaces (*netm.ag/namespaces-267*) instead of pseudo-OOP code or prefixed functions, especially when you're building themes for clients directly and know the environment the code will run in. I only learned about PHP namespaces late last year, and now I want to go back and rewrite everything I have ever done.

Defining a namespace or sub-namespace helps ensure your files are organised and readable. Each of these namespaces is a different scope, so you can have functions with the same name throughout your project. As long as these functions are in a different namespace, you will not create conflicts or cause fatal errors in the site.

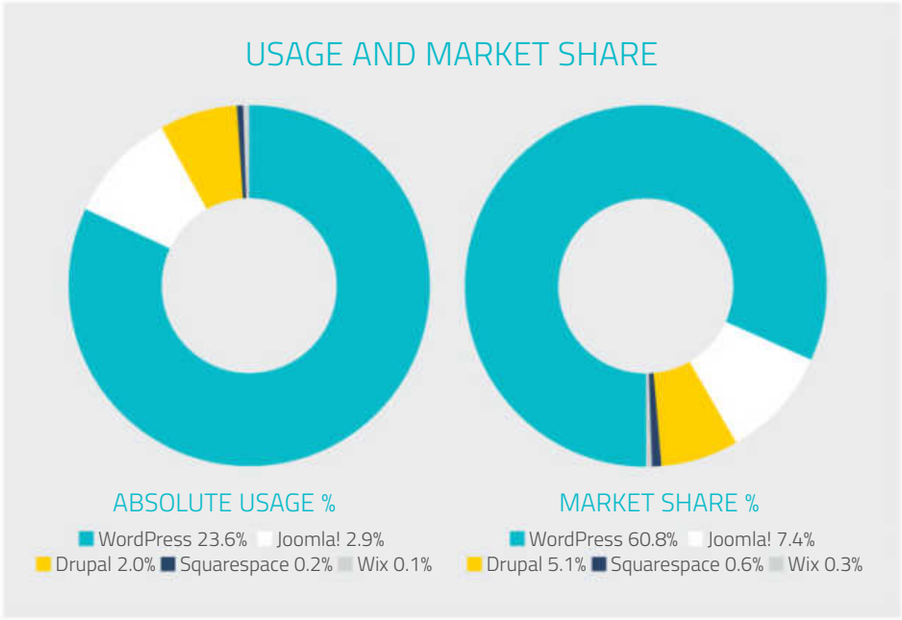
In the examples shown in the images above, you'll see that both files have functions called `load()` and `register_taxonomy()` but they are not within a class. They are written just like the procedural code we are familiar with working in WordPress. And since they are in separate namespaces there will not be issues.



Flowchart Tom McFarlin's guide helps you establish if a feature should go in a theme or a plugin (*netm.ag/mcfarlin-267*)

DO ENQUEUE YOUR SCRIPTS AND STYLES

I will be the first to admit that on the first couple of themes I built, I did this wrong. If you come from a world of static HTML, you will be accustomed to writing tags such as `<pre><link rel="stylesheet" type="text/`



Market share This chart shows the percentages of websites using the selected technologies (data: *W3Techs.com*)

`css" href="theme.css"></pre>` or `<pre><scriptsrc="myscripts.js"></script></pre>`. Years ago I saw online tutorials that said this was the way to go. They were wrong.

WordPress provides two similar functions that enable you to add stylesheets and scripts to your theme: `wp_enqueue_style()` and `wp_enqueue_script()`. Why is it important to use these functions? Because they allow you to define dependencies.

The `wp_enqueue_script()` function takes five arguments: `wp_enqueue_script( $handle, $src, $deps, $ver, $in_footer )`.

Do write procedural code. Defining a namespace or sub-namespace helps ensure files are organised and readable

The third argument is dependencies. For example, let's say you are adding a slider script to your theme and this script requires jQuery to be loaded before it will work.

Using the direct linking method, you would have to pay close attention to the order in which you placed these files. Using `wp_enqueue_script()` to include your slider script means you can pass a dependency of jQuery and WordPress will make sure jQuery is loaded on the page before your slider script.

WordPress ships with a couple of dozen scripts (*netm.ag/scripts-267*) that you can add using this method by simply calling the handle to enqueue, or passing the handle as a dependency of your script. The `wp_enqueue_style()` function also has a dependency argument, so you can determine load order of style sheets as well.

The other reason I recommend using `wp_enqueue_script()` is the last parameter in the `wp_enqueue_script()` function referenced previously, which determines whether or not the script loads in the header or the footer of the site.

By default, the function will load the script in the header. I wish it was the other way around, as most scripts should be loaded at the bottom of the page, for performance reasons (*netm.ag/performance-267*). However, this gives you an easy way to determine what loads in the header versus the footer of a site,

Theme architecture

Below you will find my current approach to architecting a theme. I am a big proponent of making my work as modular as possible, and this extends to my theme directory structure. In the structure here, I've tried to ensure it's straightforward enough that anyone could understand it.

Project Root
- Node_modules
- Bower_components
  - Fonts
  - JS
    - /src (JS you write goes here)
    - project.js (concatenated)
    - project.min.js (minified)
  - CSS
    - project.css
    - project.min.css
    - project-admin.css
    - project-admin.min.css
    - editor-style.css
    - editor-style.min.css
  - /SCSS
- Images
- Includes (PHP classes and files go here)
- Templates (page templates go here)
- Partials (template parts go here)
- Languages

SCSS (broken out since it has additional levels)
- /Global
  - functions, mixins, placeholders
  - variables
- /Base
  - reset/normalize
  - typography
  - icons
  - wordpress (a place to cover the classes that WordPress automatically generates)
- /Components
  - buttons
  - callouts
  - toggles
  - (all other modular reusable UI components)
- /Layout
  - header
  - footer
  - sidebar
- /Templates
  - home
  - page
  - page templates
  - archives
  - blog
  - (all page/post/CPT specific styles)
- /admin
  - partials as needed
- admin.scss
- project.scss
- editor-styles.scss

Template hierarchy

The WordPress template hierarchy is the system by which WordPress chooses which file is used to present information to the viewer. Below is a chart created by Rami Abraham and Michelle Schlup (see the original at *wphierarchy.com*) to visualise the hierarchy.

The chart starts on the far left, with the seven types of views that can be provided to the user. In WordPress lingo, everything is a 'post' since the framework was born out of blogging. A 'page' is a special kind of post. The first thing we have to determine is if we are looking at a singular post or a collection of posts, such as a blog archive. If we are looking at a collection, we start in the top-left corner with the archive page. We can then follow the chart to the right to determine which PHP file WordPress is going to choose to present information to the site viewer.

Are we looking at a set of posts that have been tagged 'tabby cats'? Then WordPress is going to look for files in this order: 'tag-tabby-cats.php' > 'tag-1.php' (assuming tabby cats is the first tag on your site) > 'tag.php' > 'archive.php'.

If it can't find any of those files, it will fall all the way back to 'index.php', which is the main reason every theme requires an 'index.php' template. In other systems, the index file is usually the homepage, but in WordPress it is the ultimate fallback file. It will only be your homepage if you have not included a more appropriate file for the hierarchy flow to choose.

instead of having to copy tags between the two. This brings us to our first don't …

DON'T REMOVE WP_HEAD() OR WP_FOOTER()

These are arguably the two most important hooks in a WordPress theme. You will break everything if you remove them. They basically perform the same function, but do it in different locations. The `wp_head()` hook should be included right before the closing `</head>` tag (usually in your 'header.php' file) and `wp_footer()` should be included right before your closing `</body>` tag (usually in your 'footer.php' file).

In the last 'Do', we talked about loading files in the header or the footer. These hooks are how WordPress includes these enqueued stylesheets and scripts on the page. Not only does your theme need them to function correctly, but plugin authors also use them to add stylesheets and scripts.

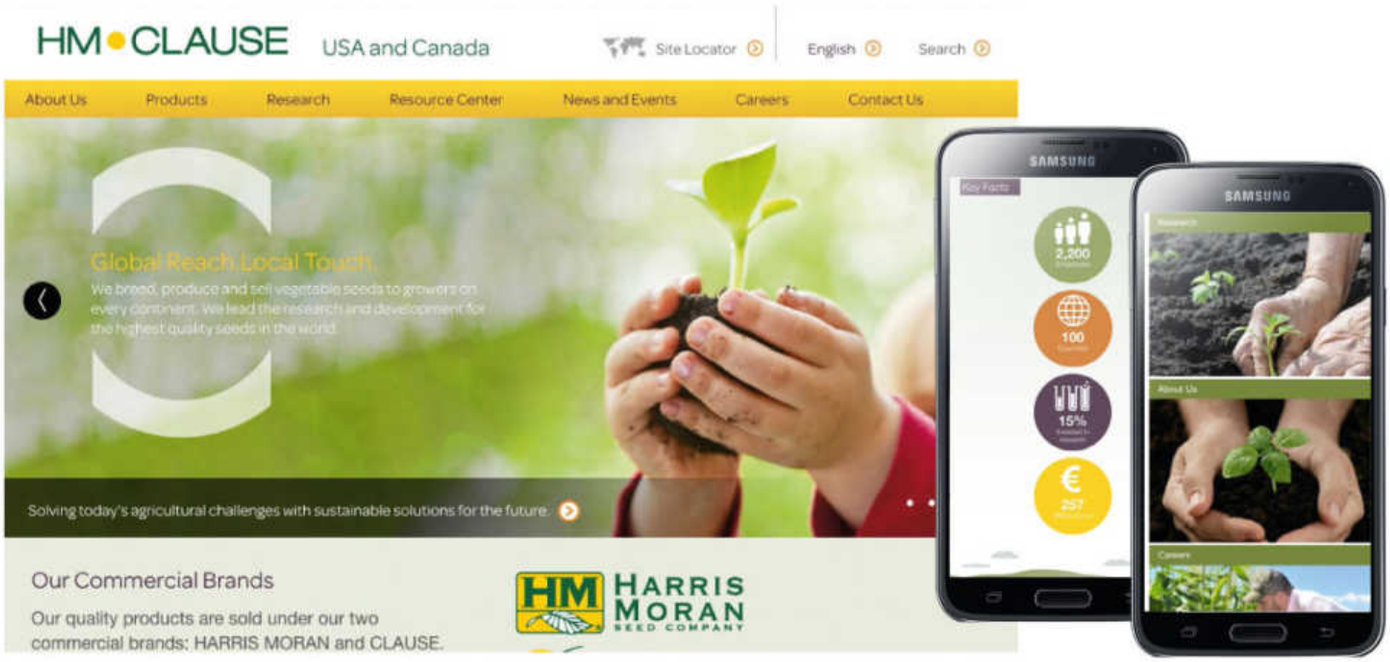
When I was a freelancer and a client would tell me a script or plugin was not working correctly, the first thing I would check for was calls to these hooks in the theme. A huge number of times, I found these hooks missing.

DON'T FILL UP FUNCTIONS.PHP

Everyone on my team knows I am very particular about how we structure and architect our projects. If your theme is simple and your 'functions.php' file is under around 100 lines or so, then you can skip to the next section. Still with us? Good.

Most of the sites I build at 10up are massive undertakings. They are the kinds of projects that involve multiple sets of hands and many months. For this reason the way we structure our work is vitally important. Our work has to make sense to everyone else on the team, but it also makes sense down the road when someone else picks it up to work on it.

I will go into more depth on file structure approach later in this article, but the basic rule here is not to cram everything into one file. This makes your code harder to read and maintain. Also, when working in a team, having concerns separated into different files makes



HM CLAUSE When working on complex projects (such as this HM.Clause site built by 10up), keeping your functions file simple is vital

collaboration a more enjoyable process. While we don't necessarily strictly follow the single responsibility principle (*netm.ag/responsibility-267*) unless we are writing a true OOP class, the aim is to maintain a 'one file equals one concern' principle.

An example functions file might look like this:

```
<pre>
<?php

/**
 * Theme functions and definitions
 *
 *
 * @package Core
 * @since 0.1.0
 */
namespace Core;

// Useful global constants
define( 'CORE_VERSION',     '0.1.2' );
define( 'CORE_URL',         get_stylesheet_
directory_uri() );
define( 'CORE_TEMPLATE_URL', get_template_
directory_uri() );
define( 'CORE_PATH',        get_template_
directory() . '/' );
define( 'CORE_INC',         CORE_PATH .
'includes/' );
define( 'CORE_ASSETS',
CORE_URL . '/assets/' );

// Include files
require_once CORE_INC . 'setup.php';
require_once CORE_INC . 'navigation.php';
require_once CORE_INC . 'settings.php';
require_once CORE_INC . 'query_mods.php';
require_once CORE_INC . 'views.php';
require_once CORE_INC . 'helper.php';
require_once CORE_INC . 'seo.php';

// Load Files
add_action( 'after_setup_theme', 'Core\
Navigation\setup' );
add_action( 'after_setup_theme', 'Core\Query_
Mods\setup' );
add_action( 'after_setup_theme', 'Core\Views\
setup' );
add_action( 'after_setup_theme', 'Core\Setup\
setup' );
add_action( 'amcn_setup_theme', 'Core\Seo\
setup' );
add_action( 'init', 'Core\Settings\setup' );
</pre>
```

Don't cram everything into one file – this makes your code hard to maintain. Aim for 'one file equals one concern'

DON'T LITTER YOUR TEMPLATES WITH LOGIC

The concept here is very similar to the last 'Don't'. Unlike other frameworks, WordPress makes it virtually impossible not to mix logic, PHP and markup. But we can be careful to use as little of this logic inside of our templates as possible.

Write functions (that you include from your 'functions.php' file) to build your logic. Include the functions in your template – or even better, include them in your template parts (we'll discuss these next). On GitHub, I've created two examples to help you. At *netm.ag/clean-267* you'll find a good, clean template. At *netm.ag/bad-267*, you'll see a bad example, in which logic has been mixed into the template.

DO USE TEMPLATE PARTS

Have you noticed that I like breaking things down into components a lot? WordPress has a highly useful function

The Loop demystified

No matter what piece of content you are showing, chances are the Loop will be involved. **Pat Ramsey** explains more

Once WordPress knows what type of content it's supposed to deliver, it makes a call to the database for the matching content. The 'Loop' is the mechanism for acting on that content.

The basic Loop looks like this:

```
<pre>
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>

<?php /* Do something. */ ?>

<?php endwhile; else : ?>
        <p><?php _e( 'Sorry, no posts matched your criteria.' ); ?></p>
<?php endif; ?>
</pre>
```

Stepping through this – first we see if we got something from the database:

```
<pre><?php if ( have_posts() ) ?></pre>
```

As long as we still have content:

```
<pre><?php while ( have_posts() ) ?></pre>
```

Let's prepare each instance of the content for use:

```
<pre><?php the_post() ?></pre>
```

We do something with each instance of content. We keep going. If we happen to find nothing, we say so.

```
<pre><?php endwhile; else : ?></pre>
        <pre><p><?php _e( 'Sorry, no posts matched your criteria.' ); ?></p></pre>
```

When we've run out of content, we stop.

```
<pre><?php endif; ?></pre>
```

All the markup necessary for displaying each instance of your content (your posts) would then follow. It doesn't matter if you're viewing a single page, a single post, a page of posts, or a page of posts in a single category ... The Loop 'loops' through what WordPress gets from the database and outputs it.

If you requested a page, WordPress makes the query for the content matching that page and returns one entry. The Loop only has to 'loop' once through the content. If you requested the blog page for a site, WordPress tells the database to get a bunch of posts. The Loop shows us each post, one after the other, on the page. It does this until the page limit has been reached. That's the Loop.

Looping The Loop shows each post on a site's blog page, up to the page limit

called `get_template_part()` that we can use in our templates to call bits of reusable and frequently used code. I often use this function inside the WordPress Loop (check out boxout opposite), as seen in this example from a 'page.php' template:

```
<pre>
<?php if ( have_posts() ) : ?>
<main id="content" class="site-content"
role="main">
    <?php while ( have_posts() ): the_post();
        get_template_part( 'partials/content',
'page' );
    endwhile; ?>
</main>
<?php endif; ?> </pre>
```

The inside of 'content-page.php' looks like this:

```
<pre>
<article class="entry">
<header class="entry-header">
    <?php the_title( '<h2 class="page-title">',
'</h2>' ); ?>
    <?php if ( ! empty( $post->subtitle ) ) {?>
        <p class="subtitle"><?php echo esc_
html( $post->subtitle );?></p>
    <?php  } ?>
</header>
<section class="entry-content"><?php the_
content(); ?></section>
</article>
</pre>
```

Many of my templates will use exactly the same logic and markup as above. By using `get_template_part()`, I can include that entire snippet with just one line of code. And if I need to change the markup or logic, I only have to alter it in one place.

DON'T TRUST USER INPUT

WordPress powers approximately 23 per cent of the internet. That means a couple of things. Firstly, that there are millions of people entering data either on their self-hosted WordPress websites or on sites hosted on *wordpress.com*. And secondly, that WordPress is often under attack by people trying to harness its reach for malicious purposes.

Sometimes users don't even realise when they are entering potentially harmful data. For those reasons, if we want to create secure code we should never trust user input. To find out more about this, take a look at this IBM post: *netm.ag/user-267*.

User input WordPress powers around 23 per cent of the internet, which makes security a big concern

I will not delve deeply into the process of validating and cleaning up users' input before it is added to the database (where WordPress stores all the content and data users enter from the admin area) – that sounds more like plugin territory to me. However, for our themers it is important that we properly ensure the data is secure before we show it to the user on the page (this is known as 'escaping' the data).

If you are building a theme and are creating metaboxes and systems for users to input data and content, I suggest you read WordPress' guidelines on data validation (*netm.ag/valid-267*) and plugin security (*netm.ag/plugin-267*) and watch Mark Jaquith's 'Theme and plugin security' talk (*netm.ag/jaquith-267*) to make sure you understand how to properly validate and sanitise user input.

On the WordPress VIP blog there is an excellent article about why we escape data as late as possible (*netm.ag/escape-267*). WordPress provides us with several helper functions we can use to make sure that the data we get from the database is the kind of data we're expecting and to remove potentially harmful HTML or scripts.

The functions I use most often are `esc_html()`, `esc_attr()` and `esc_url()` (*netm.ag/functions-267*). Remember, security is your responsibility – it isn't just the job of 'those backend folks'.

> Don't trust user input. WP powers 23 per cent of the internet – there are millions of people entering data

SUMMING UP

While there are many more points we could discuss on building themes, this list will get you a long way towards creating a clean, maintainable and secure WordPress codebase.

If you are looking to expand your knowledge even further, a great place to start would be the *Theme Review Handbook* (*netm.ag/*handbook-267) which is maintained by the *WordPress.Org* Theme Review Team. Happy theming!

ABOUT THE AUTHOR

ZAC GORDON

w: *zacgordon.com*

t: @zgordon

job: Educator, developer, business owner

areas of expertise: WordPress, PHP, JavaScript, workflow

SKILLS

GO FROM BEGINNER TO PRO WITH WORDPRESS

Zac Gordon presents a self-paced curriculum that will guide you on your way to becoming a WordPress power user or developer

Many people who work with WordPress have taught themselves everything they know about content management and web development. It is perfectly possible to learn how to become a WordPress power user or developer largely using free WordPress and web design sites. Online or in-person courses can also help accelerate, focus and structure the learning process.

A WordPress power user is someone who can manage content, get around the admin area, and configure themes and plugins. A WordPress developer works at the code level and can build custom WordPress sites, add advanced functionality with plugins, and possibly even develop plugins.

In this article, I will present a self-paced WordPress curriculum. I'll walk through the skills you need to become either a WP power user or a developer, and the resources you can use to get there.

Education Zac has taught high school and college students, non-technical users, and career-changing professionals how to learn WordPress

POWER USER SKILLS

A WordPress power user can do, or figure out, most things in WordPress that do not involve code. A list of specific skills might include the following:

- Adding and editing content and media
- Managing menus and widgets
- Finding and configuring plugins and themes
- Managing users and comments

Mastering these skills will enable you to manage a WordPress site on an long-term term basis, without the need to hire someone else to take care of common tasks. A power user can also likely build simple WordPress sites using existing themes and plugins. Most power users will pick up a little CSS and can do light customisations of a theme's style.

When learning WordPress, you want to have at least one practice site where you can poke around without fear of messing up a live site. Many hosting options exist that let you easily install WordPress with just a few clicks. You can also run WordPress locally on your computer using DesktopServer.

The admin area

Take the time to click through every page in the admin area. If you don't understand what something does, the WordPress Codex offers helpful documentation – just search for WordPress administration screens. Once you've completed this

Admin area A primary goal for power users is to know all of the pages, settings and options in the admin area

exercise, you'll have covered almost half of what you need to know as a WordPress power user.

Give yourself a long afternoon, or spread it out over a couple of days. If you're focused, this process can take just a couple of hours.

Custom content

WordPress comes with two default content types: posts and pages. Plugins and themes can also be used to add custom post types and fields for things like products, projects, team, slides and locations. Essentially, any content type that does not fit the posts and pages model.

A power user does not necessarily need to know how to set up custom post types. However, you should prepare yourself to see custom post types in the admin navigation. For information on how to manage and use specific custom post set-ups, refer to the documentation for the theme or plugin that created the post type.

Plugins and themes

Once you know how to manage content and admin settings, you will want to master the skill of finding and configuring plugins and themes. Include the words 'WordPress plugin' in a search, along with the particular functionality you want (such as 'WordPress calendar plugin'), and you will get results for both free and paid plugins across a range of sites. When searching for a theme, try to use specific, niche terms, such as 'Legal WordPress themes' or 'Events WordPress themes'.

When reviewing a plugin or theme, there are a few things to look out for. These include ratings, the support provided, the number of downloads, and if it's regularly updated. You also want to get a feel for if it's coming from a reputable place.

Plugins tend to create a settings page in the admin area, and themes are starting to put their theme options in the Customizer view. Refer to the documentation for the particular plugin or theme for details on how to set up what you need.

IN BRIEF

A SUMMARY

I've covered quite a bit of ground in this article, so this boxout summarises the aims, timeframe and key resources for becoming a WordPress power user or developer.

The skills

Power user

Knows how to use the admin area and set up plugins and themes. You can learn these skills quickly, without knowing code. Allow around two to four weeks.

Developer

Can build and customise themes and possibly plugins. Requires a more significant learning curve and some basic web development skills, like CSS, JS and PHP. Allow around one to three months.

Where to learn

You can get quite far from just using the WordPress Codex (*codex.wordpress.org*) along with some support from online learning sites like Codecademy or Treehouse. Here are a few resources:

Power User

- Practice administering content and settings on test site
- WordPress Codex admin screen pages (*netm.ag/adminpage-271*)
- WordPress Codex for menus (*netm.ag/menus-271*) and widgets (*netm.ag/widgets-271*)
- Codecademy for HTML and CSS (*codecademy.com*)

Developer

- WordPress Codex - Template hierarchy (*netm.ag/hierarchy-271*)
- WordPress Codex - Template functions (*netm.ag/temp-271*)
- WordPress Codex - Actions (*netm.ag/typical-271*) and filters (*netm.ag/filter-271*)
- Codecademy for HTML, CSS, JavaScript, PHP
- Treehouse WordPress development track (*netm.ag/track-271*)

Apply your skills

A number of niche jobs exist within the WordPress ecosystem for power users and developers. Power users can often work managing websites for clients either part-time or full-time. They are also able to build simple sites from scratch for themselves or for clients.

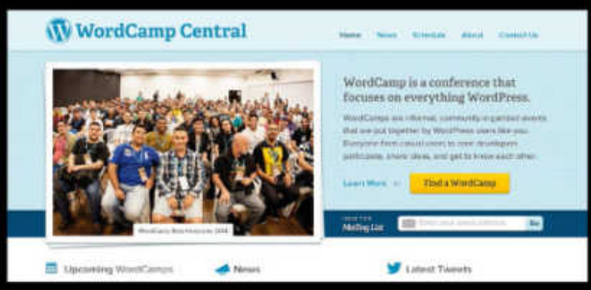
WordPress developers often work for themselves or a company doing client work on themes and plugins. Some WordPress developers decide to take the product approach instead, and build, sell and support their own themes or plugins.

FOCUS ON

LEARNING TIPS

There are many different approaches to teaching yourself WordPress. Here is a run-down of my top tips:

- Have a demo/test site you can experiment with
- Try to understand on your own what all the settings do
- Practice importing content as well as adding it manually
- Set clear goals for what you would like to learn
- Schedule time on a regular basis to study
- Connect with other WordPressers at meetups and WordCamps
- Build a network of more advanced WordPressers who can answer your questions
- Take time to read through the WordPress Codex
- Checkout WordPress tutorial sites and *WordPress.tv*
- Try a formal in-person or online WordPress course
- Practice building different types of WordPress sites with themes and plugins
- Try coding your own custom theme from scratch using the Underscores starter theme
- Read up on how child themes work
- Practice making child themes for several existing themes
- Download and explore the WordPress plugin boilerplate
- Read through the code of simple plugins to learn how they work
- Take time to scan the WordPress core files
- Set up a local development environment using DesktopServer or Vagrant
- Use phpMyAdmin or the command line to explore the WordPress Database schema



Meet up Attend WordPress events like WordCamps to learn more about WordPress and meet other users and developers in your area

Even if a plugin or theme looks great during the review process, it may not work as you need it to once it's installed. You may have to try several comparable plugins or themes before settling on the ones that work best for a specific project.

Since each plugin and theme requires a different configuration process, you should try building a few different practice sites. For example:

- A multi-author blog / magazine-style site
- A professional business brochure or portfolio site
- An ecommerce or members-only subscription site

These projects will expose you to a variety of themes and plugins, each with slightly different set-up processes.

Customising sites with CSS

With CSS, you can make customisations that are not possible from within the Customizer. For example, CSS can rearrange elements, hide things, change colours, adapt font sizes, apply mobile styles and much more.

It only takes a day to learn the essentials of CSS (and HTML) on Codecademy, and it's free (*netm.ag/academy-271*). You don't have to remember everything, just the basic structure, which is pretty simple. Plugins like Jetpack (*jetpack.me*) let you easily add your own custom CSS to any theme.

DEVELOPER SKILLS

WordPress developers can modify WordPress to meet just about any need, from simple blogs and brochure sites to social networks and ecommerce sites, or even web apps and APIs. A WordPress developer should know the basics of the following:

- Local development and migration
- Theme customisation and custom site design
- Template hierarchy and template tags
- WordPress hooks and the REST API

Learn online Online resource Codecademy has some excellent tutorials to help you learn core development languages

This may seem like a lot (it is), but many WordPress developers evolve their skillset over a period of months or years, as they take on more complex and varied projects.

The path to becoming a WordPress developer can take a bit more time and effort than is required for a power user. One reason for this is because it involves familiarising yourself with some of the core languages of web development.

LEARN THE LANGUAGES

If you do not know the basics of these languages already, set aside a week or a couple of solid days to work through the HTML, CSS, JavaScript and PHP courses on Codecademy.

You will not need to know a lot about databases at first, but you should definitely familiarise yourself with the Moving WordPress page in the WordPress Codex (*netm.ag/move-271*) for information about site migration. The hosting company SiteGround also has some good tutorials on how to use the popular MySQL admin tool phpMyAdmin (*netm.ag/admin-271*), which you may need.

WordPress developers can modify WP to meet almost any need, from websites to apps

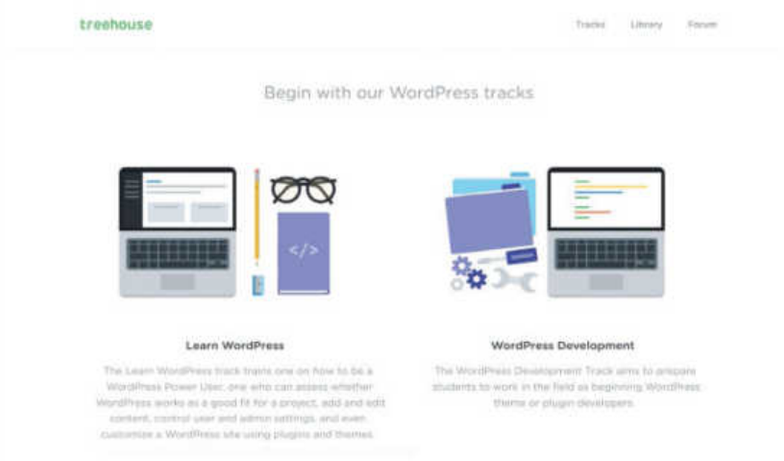
Theme development and customisation

WordPress themes contain a list of common core files. Theme developers customise these files and add additional custom files as needed for each particular site.

Check out *wphierarchy.com* to see what is referred to as the Template Hierarchy – the full list of default template files. Template files contain template tags – PHP functions specific to WordPress that enable you to access information about the WordPress install, display content, and include other files. Take an hour or two to browse through the functions listed on the Template Tags page in the WordPress Codex (*netm.ag/tag-271*).

Next, download the starter theme Underscores (*underscores.me*) from the folks at Automattic, and explore the files it comes with. This includes all the core PHP template files, but no CSS. Try using this theme as a codebase upon which to build your own custom theme.

If you want to learn WordPress theme development in a condensed amount of time, I suggest the WP developer track at Treehouse (*netm.ag/track-271*).



Treehouse track Learning sites like Treehouse offer online courses to help you learn WordPress development quickly and efficiently

Programmable WordPress

Hundreds of hooks exist in WordPress that let you override default behaviour or add your own code when certain events take place. Look up 'Actions run during a typical request' on the Codex (*netm.ag/typical-271*) to see hooks in the order they get called as a page loads on the frontend or backend of the site. The Codex also lists all the filter hooks that enable you to modify content and settings.

Take time to browse the actions and filters sections of the Codex so you know what is possible. Then read one or two of the many blog posts that explain how to use actions and filters.

The WordPress REST API promises to open up WordPress to interactions that do not use the admin area or typical frontend themes. Familiarise yourself with *wp-api.org* and the REST API talks on WordCamp TV (*netm.ag/REST-271*).

PUT YOUR SKILLS TO WORK

One of the things that excites me about WordPress is how quickly you can develop employable skills. For example, the ability to update content is on its own an employable skill for a site editor, offering someone the chance to earn residual income with just a few clicks and copy and paste. A power user can even bridge into the business of building simple sites, charging significant money for their work.

Most tech job boards have regular listings for freelance and full-time WordPress developer positions. Local WordPress meetups and WordCamps will help plug you into more opportunities as well.

As an educator, I will close with a reminder that as you learn WordPress you will begin to know things that other people do not know yet. Seek out opportunities to help speed up the learning process for others, whether as a paid trainer, a speaker at WordPress events, or as someone who answers others' questions online.

WORDPRESS

WORDPRESS EVOLUTION

Marko Heijnen reflects on the 4.0 update and considers the future for WordPress

WordPress recently launched a new version, 4.0, amid much excitement around the release's updates and how they would benefit end-users and developers. The release boasts over 1,200 updates since 3.9. Although no groundbreaking new additions were included, 4.0 leads the way for future releases, with a number of underlying changes and clean-ups making WordPress 4.0 a more stable CMS.

NEW ADDITIONS

My personal contributions to the release of WordPress 4.0 included one patch. I initially co-created the `WP_Image_Editor` class with Mike Schroder in WordPress 3.5. With ongoing improvements, the update for WordPress 4.0 adds the method `get_quality()`. An important class that handles all image manipulation within the platform, the new method will help developers retrieve the current quality of an image when using the class directly in their own projects.

Whilst a number of incremental changes are present within the frontend and backend, the WordPress 4.0 release additionally included code cleanups and an application of standards for increased code quality. For example, almost all uses of the PHP function `extract()` were removed. The function previously exploded an array into variables, which although handy to use, overwrote variables without users knowing. Users were previously unsure as to what was being parsed by `extract(`. The removal of the function increases the stability of WordPress as a package, as no variables have the possibility of being overwritten.

> 4.0 leads the way for future releases, with underlying changes and clean-ups making WordPress 4.0 a more stable CMS

Additionally, Scrutinizer, a code inspection platform, was used to detect dead code and unused variables by searching through the codebase to find possible issues. Although not necessarily fixing every issue, the access modifiers gave a good indication as to the current status of the code, increasing the overall quality.

The code cleanups and removal of dead code within update 4.0 are a huge benefit to developers, making it easier to contribute to the ongoing improvement of WordPress as a CMS platform.

With `textarea` support for Customizer, less code is required for developers to test themes. When installing plugins on a dev version of WordPress, a new tab is visible. This saves developers time when completing updates as it displays all features being worked on as plugins.

THE FUTURE

The most notable change in this release is the creation of updates we can build on. But what do I think is the next big step for WordPress? The incorporation of a RESTful API.

More complex websites and mobile apps are continually being built with WordPress, so it is important for WordPress to have a RESTful API. Such an inclusion could support current WordPress apps as well as mobile apps that use WordPress as a backend. As a feature available as a plugin now, users are currently able to test this service.

Going forward, the platform will continue to update its legacy code and add the APIs that developers need to better accommodate these user-cases as their popularity increases. Although a lot of work is still needed around this, I am hopeful that we will see this update in WordPress 4.1 or 4.2.

Secondly, I expect to see the rebuild of the image editor in WordPress. A new project in which I am involved, the current version displays a number of issues including extendibility. Although the status of the build and release date are currently undetermined, I see a huge potential here for a more extendable UI for plugin developers to create new features for users.

There are also plans to extend the class `WP_Image_Editor` with new methods. Hopefully these extensions add the default inclusion of image filters (such as greyscale or sepia). As a personal goal for this project, I will begin working on new ways to show images on mobile.

PROFILE

Marko (*markoheijnen.com*) is an avid WordPress enthusiast, and has been making contributions since 2010. He recently launched his own agency, CodeKitchen

SUBSCRIBE TO NET

GET THE NO.1 CHOICE FOR WEB DESIGNERS AND DEVELOPERS DELIVERED TO YOUR DOOR, YOUR DEVICE, OR BOTH

PRINT EDITION ONLY

Take out a print subscription to **net** and get your copy before it hits the shops. Each issue is packed with the latest web trends, technologies and techniques

FROM
£27.50
BASED ON A 6-MONTH SUBSCRIPTION

SAVE UP TO
37%

DIGITAL EDITION ONLY

Take out a digital subscription to **net** for on-the-go access to our fully interactive edition, with streaming screencasts, extra images and more

FROM
£22.50
BASED ON A 6-MONTH SUBSCRIPTION

SAVE UP TO
45%

Terms & conditions: Prices and savings quoted are compared to buying full priced UK print and digital issues. You will receive 13 issues in a year. If you are dissatisfied in any way you can write to us or call us to cancel your subscription at any time and we will refund you for all un-mailed issues. Prices correct at point of print and subject to change. For full terms and conditions please visit: *myfavm.ag/magterms*. Offer ends 31/12/2015

NOT IN THE UK?
TURN TO PAGE 59

NEW PRINT & DIGITAL EDITION

Enjoy a combined print and digital subscription, and take advantage of print as well as exploring the fully interactive digital experience

GREAT REASONS TO SUBSCRIBE

- Print edition delivered to your door
- 13 issues in a one-year subscription
- iPad and iPhone edition download
- Android edition download
- Money-back guarantee

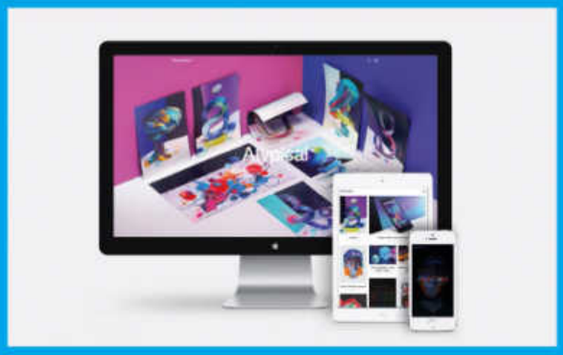
FROM
£33.00

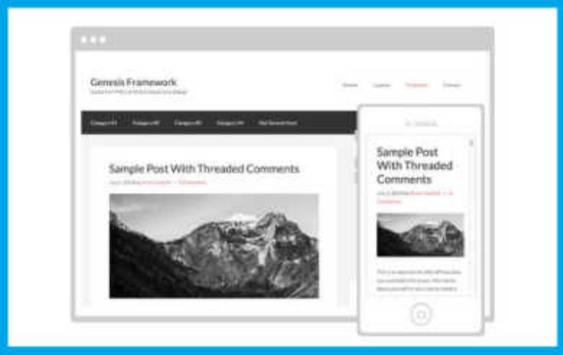
SAVE UP TO
59%

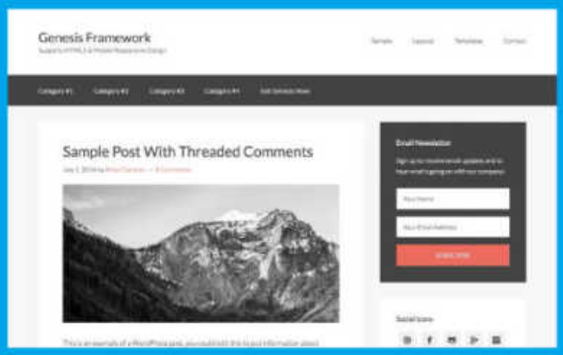
myfavouritemagazines.co.uk/NETSUBS15

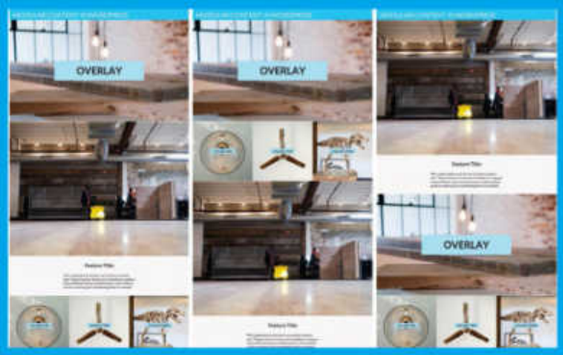
Subscriptions

PROJECTS









ABOUT THE AUTHOR

SHANNON SMITH

w: *chroni.ca*

t: @cafenoirdesign

job: Founder, Café Noir Design

areas of expertise: Multilingual web development, accessibility

PERFORMANCE

8 WAYS TO OPTIMISE YOUR WORDPRESS SITE

Shannon Smith walks through eight things you can do to speed up your WordPress site and keep your users coming back

We've got you covered when it comes to WordPress themes, and you know you can count on us for a great WordPress tutorial or two, but what if you want to make your WordPress site perform better? There's plenty you can do to boost your site's performance – check out these pro tips!

1 CACHE YOUR SITE

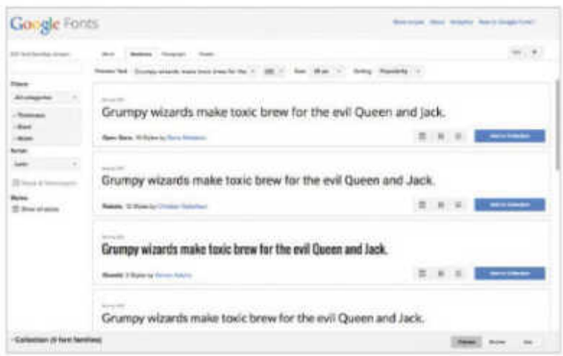
Even if you do nothing else, install a caching plugin like WP Super Cache or W3 Total Cache. This will serve your dynamic WordPress site, as if it were static HTML content. Instead of regenerating a page for each visitor, your site will be generated once by the PHP engine.

Enable server-side caching. Use zlib and apache's '.htaccess' file to compress your files using gzip. Your server needs to have zlib installed and mod_deflate enabled. You can also enable browser caching. Set appropriate expires headers and disable entity tags (eTags).

2 HOST FILES EXTERNALLY

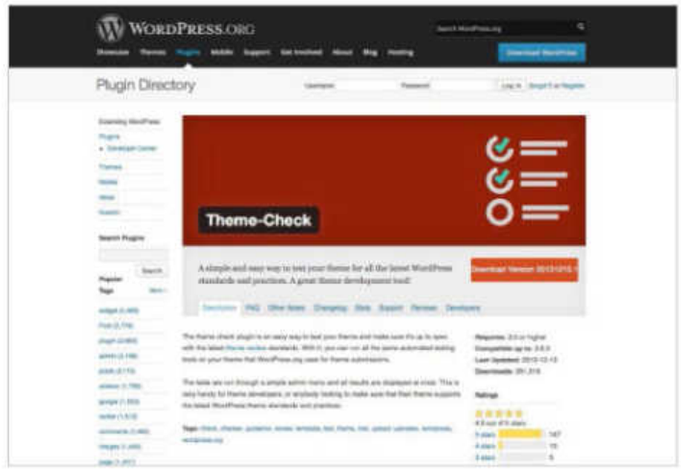
Try to host some of your static files externally. Browsers usually only download two files in parallel from a single domain. Using multiple domains increases the number of files that can load at any one time.

Consider using externally hosted fonts, like Google Fonts. Use a content delivery network (CDN), such as Amazon CloudFront (*aws.amazon.com/cloudfront*), to serve your static files from multiple servers at the same time. If you use the Jetpack plugin, you can use its Photon service to host your image files on the *WordPress.com* network for free.



Left Install a caching plugin like WP Super Cache

Right Consider using externally hosted fonts, like Google Fonts



If hosting files externally isn't an option, it is possible to host static files in a secondary domain or even a subdomain, with the added benefit of serving files from a cookie-free domain, which also speeds load times.

3 REMOVE PLUGINS

Could your plugins be slowing down your website? The P3 (Plugin Performance Profiler) will let you know the effect of each installed plugin on your website's performance. Delete anything that is unnecessarily inefficient.

Ensure your CSS is as lean as possible and reduce the number of classes in your code

It's also a good idea to delete any plugins that you aren't using. Even when deactivated, extra plugins don't just take up file space on your server, they can also fill up your database and leave extra files and functions that can slow down your site.

It's important to remove plugins through the WordPress Admin panel, when possible. Good plugin developers often add clean-up routines that remove extra tables from the database, and delete files and functions, but these only run when you remove the plugins properly.

Left The P3 (Plugin Performance Profiler) can let you know the effect of each installed plugin

Right Use the WordPress Theme-Check plugin to ensure your code is up to standard

4 OPTIMISE THEMES

Just like plugins, your choice of theme can significantly affect your website's speed. If you are using a prebuilt theme, try to find one that has been optimised for mobile, accessibility and SEO, as these optimisations also generally lead to faster, leaner themes.

If you are building a custom theme, always use valid (don't forget to check) semantic code, and avoid broken links and missing files. Use the WordPress Theme-Check plugin to ensure you respect the latest WordPress theme review standards. Some of these standards help your theme load faster.

Use CSS in place of image files where possible. CSS Hat (*csshat.com*) is a tool that can help you turn Photoshop files into CSS. Create image sprites for your design images. Limit your PNG files to only the colour palette in use, and try to avoid filters like alphaimageloader.

Ensure your CSS is as lean as possible and reduce the number of classes in your code. When using javaScript, including jQuery, ensure you are only calling each script once. Use the smallest number of CSS and JavaScript files possible.

Make sure you use the `wp_enqueue_style()` and `wp_enqueue_script()` functions to call your CSS and JavaScript files, which automatically minifies and concatenates your CSS and JS. You could also minify your CSS and JS using a minify plugin. If you are using one of the default scripts included and registered by WordPress, use the already minified version, when available.

You might also consider using one of the scripts in the Google Hosted Libraries, so you don't have to host it yourself. Be aware that these may not work with all plugins, and you will also have to de-register WordPress' original copy in some cases, to avoid calling scripts twice. You should also use `wp_enqueue_script` with externally hosted scripts.

Reduce the number of php queries. Hardcode whatever you can. Store your functions as a variable or in an array, so that the same function is only called once. Remove any unused code from your

'function.php' file. And don't forget to debug with WP_DEBUG, among other tools.

5 FINE TUNE YOUR CONTENT

You can speed up your site by reducing the amount of content on the page and in your feeds. Put your sidebar(s) on a diet and reduce the number of widgets you use. Remove unused widget areas from your theme. Reduce the number of posts on each page and show excerpts.

Be careful about using native social network services, ad networks services, or any service that tracks users on your site in real time. Compare their speed to that of alternative plugins that serve similar functions.

If budget allows, you can see improvements in speed with dedicated virtual hosting

Optimise and compress your images before uploading. There are plugins that can do this on upload, like Imsanity (*netm.ag/imsanity-271*). You can also use the Yahoo Smush it service, through a plugin like WP Smush to further compress images.

Be sure to properly set the default image sizes in the WordPress admin, so that the browser won't need to scale your images on the fly. You may need to create additional custom image sizes in your 'functions.php' file. You can also use the Lazy Load jQuery plugin to delay loading images outside the browser viewport. There are several WordPress Lazy Load plugins.

6 MAINTAIN YOUR DATABASE

Make sure you optimise and repair your MySQL database. You can use a plugin, or do this through a database tool like phpMyAdmin.

You can also do several things to lighten the load on your database: empty your WordPress trash and spam, delete inactive users, reduce the number of revisions that WordPress stores for your posts and pages. Delete unused images.

7 CHOOSE A GOOD HOST

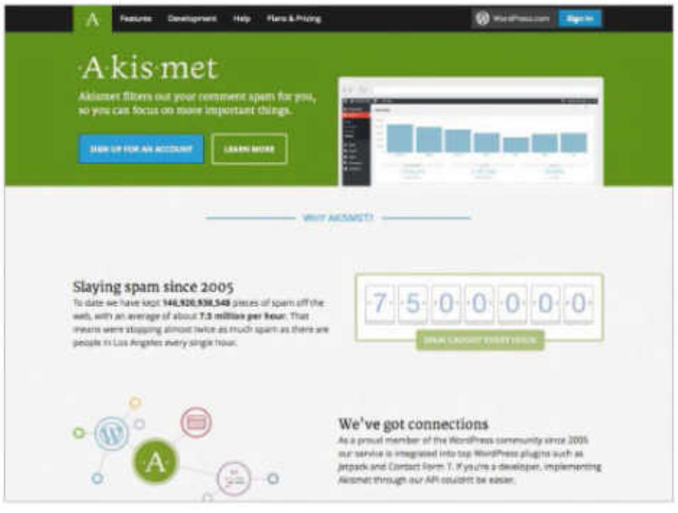
You get what you pay for when it comes to hosting. Choose a good host with a server near your target audience. If your budget allows you to move away from shared hosting, you can also see improvements in speed with dedicated virtual hosting (though you will need to know how to manage your server space to really optimise your resources). And make sure that your host allows your account to have sufficient memory and zlib compression tools.

8 TAKE CARE OF SECURITY

It won't matter how much you optimise your site if your site is hacked or under attack. Keep WordPress and your themes and plugins up to date. Use a host with good security. Block spambots using an anti-comment spam plugin like Akismet or Bad Behavior, and delete your comment spam often. Use a plugin, like Login LockDown, the '.htaccess' file to reduce brute force attack attempts. Disable image hotlinking from your site using '.htaccess'.

IS IT WORKING?

Don't forget to check your work. Google's PageSpeed insights can give you a good idea of what needs improvement. Other tools include Yahoo! YSlow and Pingdom Website Speed Test.



Left CSS Hat is a tool that can help you turn Photoshop files into CSS

Right Use an anti comment-spam plugin like Akismet

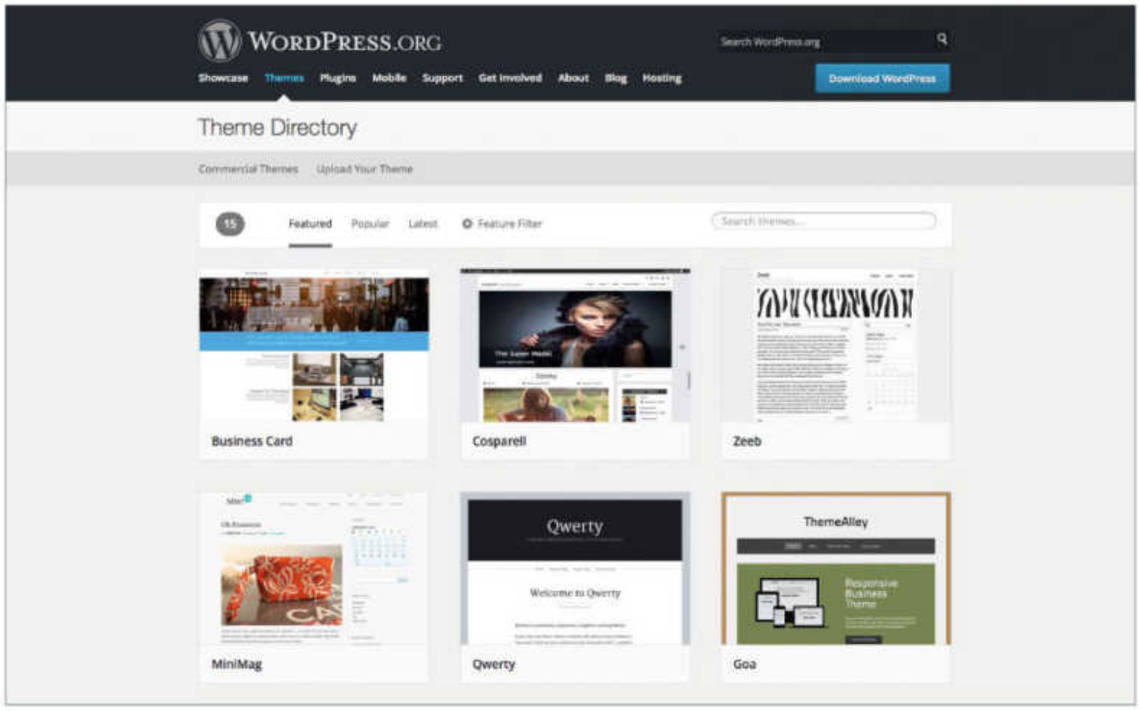
ABOUT THE AUTHOR

JAMES KOSTER

w: *jameskoster.co.uk*
t: @jameskoster

Job: Designer and developer, WooThemes

Areas of expertise: WordPress theme/ plugin design and development



FREE THEMES

EXPLORE WORDPRESS' FREE THEME OPTIONS

All WordPress-powered websites use a theme. **James Koster** looks at why a free theme might be a better choice than a commercial one

You probably already have a good picture in your head of what a WordPress theme is. But have you ever specifically considered the intended purpose of a theme, or why themes were introduced way back in WordPress 1.5? *wordpress.org* describes a theme like so:

> *Fundamentally, the WordPress theme system is a way to 'skin' your weblog. Yet, it is more than just a skin. Skinning your site implies that only the design is changed. WordPress themes can provide much more control over the look and presentation of the material on your website.*

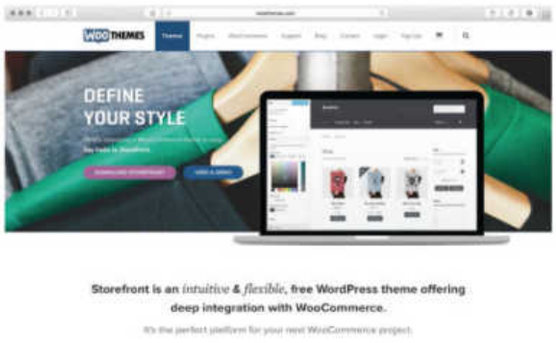
The key word in there is 'presentation'. A WordPress theme is intended to provide the frontend interface for your website's content. That's the crux of it. Of course, they can do more.

WordPress is way more than a mere blogging platform. Programmatically, there's nothing to stop you building an entire ecommerce platform into a single theme. But that's a terrible idea. The fact is that anything beyond presentation is better suited to a plugin. In short: themes are for content presentation, plugins are for content creation.

It sounds obvious, right? But I'm going to bet that everyone reading this article has probably used or built a theme that includes features and functionality beyond of the intended scope of what a WordPress theme is expected to provide. That's nothing to be ashamed of. It's not even necessarily a bad thing. But it is fundamentally wrong, and things that are

RESOURCE

Theme Hybrid is a project run by Justin Tadlock. Using the Hybrid Core framework, the members build a variety of themes that are uploaded to *wordpress.org: themehybrid.com*



In store Storefront is the official theme for WooCommerce. The core theme is lightweight and extensible, offering various presentation options

fundamentally wrong have a habit of coming back to haunt you.

Having read this opening section you may think I'm being pedantic. You may even think I'm crazy. But hopefully after reaching the end of the article you'll feel some sense of enlightenment.

THE STATE OF PLAY

Currently, commercial theme marketplaces like ThemeForest (*themeforest.net*), Creative Market (*creativemarket.com*) and Mojo Themes (*mojo-themes.com*) are thriving. They dominate the landscape and it's easy to see why. On the surface, they offer a win-win proposition for theme creators and end users alike: consumers get access to 'professionally'

On the surface, commercial themes marketplaces offer a win-win proposition

built products for a fraction of the price of hiring a designer or developer directly, while theme creators can operate in a contextually rich market to promote and sell their work.

Commercial themes have been big news since before the turn of the decade, with many authors earning tens of thousands a month selling them. This lucrative opportunity has attracted more and more authors to the various marketplaces. They've become saturated with authors who now work tirelessly to distinguish their products from the competition. The obvious way for them to do this is to offer more value for money, but since the marketplace administrators control the product

FOCUS ON

FREE VS COMMERCIAL AT A GLANCE

Commercial themes offer oodles of features: page builders, sliders, portfolio modules, contact forms and so on, as a way of grabbing the consumer's attention and suggesting good value for money. But is it really worth it?

The more features a theme includes, the more it strays away from the true WordPress theme purpose. It also increases security risks, becomes more difficult to support and can have a negative impact on performance.

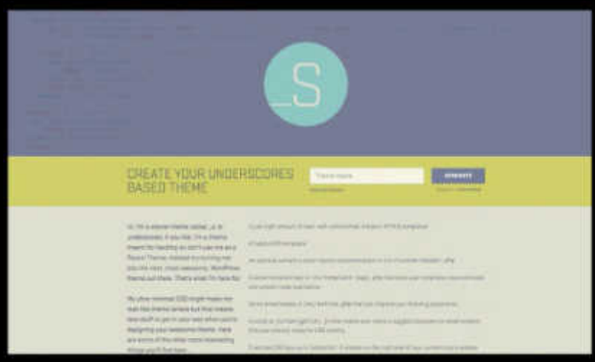
Authors of free themes generally pay more attention to things like security and accessibility, in accordance with the guidelines required by *wordpress.org*'s theme review team. They leave out content creation features that folks can add themselves later via plugins. This modular approach gives you the flexibility to easily enable/disable the features you want.

Commercial theme authors are making a lot of money, with some themes selling more than 100,000 times. It's a very lucrative space. But that is not to say that successful business models cannot be employed around free themes. Many free theme authors are successfully monetising projects by offering commercial add-ons, or premium support services. Where should you look for free themes? *wordpress.org* is a great place to start as you know all the themes there have passed a strict review.

Theme Hybrid Offer a range of themes all built to utilise their 'Hybrid Core' framework. They also offer a suite of plugins that many themes are integrated with.

IN-DEPTH

UNDERSCORES



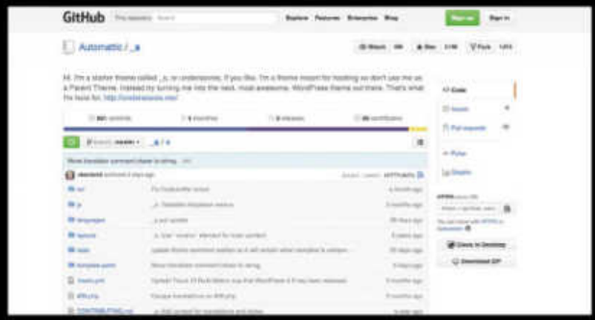
Solid base Underscores is an Automattic project designed to give theme authors a rock-solid foundation for new parent themes

Underscores (*underscores.me*) is a lightweight project that serves as a fantastic starting point for theme authors. It offers just the right amount of features out of the box, so you can build upon these safe in the knowledge that at its core, your theme will be up to speed on all modern theme development best practises.

What Underscores doesn't do is provide any design or presentational CSS (beyond some very basic responsive menu treatments). It's a starter theme that designers and developers can use as a beginning for their new projects.

Many *wordpress.com* and (more recently) *wordpress.org* themes are based on Underscores. This is helpful for agencies that make the conscious decision to use themes from *wordpress.org* where possible. Code consistency makes developers' jobs much easier and helps create a standard that others can follow.

The Underscores project is managed on GitHub which makes it easy for the wider WordPress community to get involved in its development. It also means that, with so many people contributing, it is constantly growing and improving. Many free, open source themes are developed this way – it's possibly one of the biggest advantages they have over their (mostly) closed source commercial counterparts.



Underscores Development is managed on GitHub. Developers report / fix bugs, discuss code and contribute enhancements on a daily basis

prices, authors have no choice but to add more features to boost their product's value prospect.

And before we even realised it was happening, we're in a situation where, to differentiate themselves from the competition, theme authors are executing development practises that are in direct conflict with what we outlined in the original section of this article. They are bundling 'plugin territory' features into the themes themselves. Without searching far, you can find commercial themes that include things like sliders, page builders, shortcodes and contact forms.

It's easy to see why consumers would be attracted to these products. They offer so much for so little. What's not to like? Well, we'll get to that shortly. But first, let's take a look at what is going on at *wordpress.org* – the official, open source WordPress themes directory.

THE WORDPRESS.ORG STANDARD

Many themes on *wordpress.com* were originally built for and uploaded to *wordpress.org*. For a theme to be approved for download on *wordpress.org* it must go through a rigorous review process to ensure it meets the required standards. These standards are set by folks who are regularly involved

Wordpress.org will reject any theme that includes content creation features

with the development of WordPress core itself, so we can assume they are intended best practises. In other words, they are recommended for all theme authors, irrespective of where the theme is intended to be distributed.

One of the most prominent points in these guidelines is that themes categorically must not include any content creation features. The theme review team is very strict about the fact that any theme that includes functionality that can be deemed 'plugin territory' will not be approved.

Obviously a theme that includes sliders, page builders and contact forms (the likes of which you might find in a commercial marketplace) would not be approved. You might think this makes the themes available here 'worse', but it really doesn't. It makes them more lightweight, more succinct and more flexible. There are many other important guidelines that themes must adhere to regarding things like security, consistency of theme options

and accessibility – all important topics that most marketplaces do not enforce.

However, the choice for consumers is still not clear-cut. At a glance, on one hand they're facing the free themes on *wordpress.org* that are simple in terms of features but are built in accordance with a rigid set of guidelines. On the other hand there are themes on the commercial marketplaces that offer a whole bunch of 'extra value' for a price that is so low you hardly even have to think about it.

THE COMMERCIAL CONUNDRUM

As we've already outlined, the arguments for using a commercial theme are strong. But the theme space is slowly and surely evolving. Consumers are starting to learn (sometimes the hard way) that the value proposition offered by commercial themes might not be as good as they first thought.

In the beginning of the article I suggested how following practices that are fundamentally wrong can come back to haunt you. Well, let's take a look at why you might run into problems with some commercial themes.

Portability

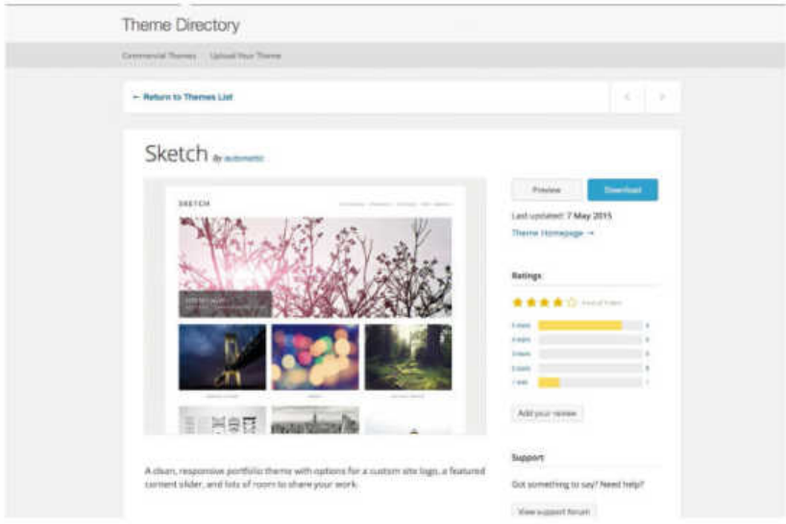
As many commercial themes include features that enable content creation (sliders and portfolio modules being two popular examples), any content created by that feature is tied to that theme, locking you into using it.

This might not seem like a problem at first, but it's unlikely that a website will use the same theme forever. Two or three years down the road you or your client might want to change the design of their site. At that point all of the custom content will be lost and will need to be recreated.

Imagine having to do this for all of your slides, services, testimonials, portfolio items – it's a real pain. Now imagine being forced into making a theme switch, for a dozen websites at once, because of a security vulnerability in a theme ...

Security

The bigger a product becomes, the more potential points of failure there are. This is made worse still in WordPress themes, as many bundle third-party PHP scripts and JavaScript libraries. If a vulnerability is found in one of those components, the site owner finds themselves at the mercy of the promptness of the theme author, who has to patch the script and upload a new version for you to download. To make matters even worse, many commercial themes do not provide update notifications, so the chance of you even learning about a critical update could be slim.



Portfolio theme
One of many themes created by Automattic, Sketch is a portfolio theme that leverages Jetpack's Portfolio module

This issue has reared its ugly head a few times throughout commercial theme history, notably with the `timthumb.php` image manipulation script, and most recently towards the end of 2014, when ThemeForest announced a vulnerability in Revolution Slider, a plugin that over 300 themes were bundling as a part of their package.

Consistency

WordPress works hard to provide a consistent experience to all users – the Customizer is a perfect example of this. But instead of embracing this API, many commercial theme authors have designed and built their own options frameworks. This means that switching themes will often require learning a whole new options layout/interface. This is frustrating for the individual user and harmful to the community as a whole.

Longevity

For a long time commercial themes have been seen as an easy and lucrative business model. But not all authors are successful. Many invest huge amounts of time and effort in their products only for them to see minimal sales. With a poor ROI they've possibly cut their losses and moved on to pastures new.

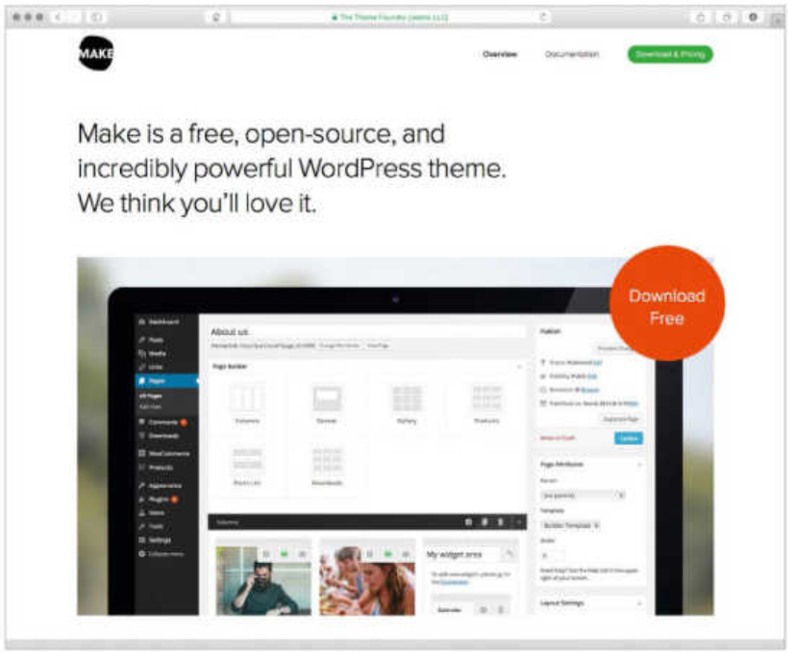
The problem for the consumer here is that while the author has left, the product remains. So anyone who purchases it might receive questionable levels of support, if any, and no updates even if a critical vulnerability is found.

Conflict with best practises

We've already covered this to an extent but it's so important right now that it's worth repeating. Commercial themes are not bound by the same guidelines as themes on *wordpress.org*. So the ▶

RESOURCE

ThemeShaper is a good blog to subscribe to if you're interested in WP themes. There are regular updates on new themes, as well as theme development posts: *themeshaper.com*



Money making Make is a great example of a free theme – the project is monetised by a commercial plugin

vast majority of authors simply don't spend the effort adhering to them.

FREE IS THE FUTURE

As commercial marketplaces slowly adopt the *wordpress.org* guidelines in an effort to ensure their authors are building better products, the themes on sale will become more akin to their *wordpress.org* counterparts, and consumers will come to understand that baked-in features do not necessarily represent value.

Free themes available on *wordpress.org* will always have the edge. Why? Portability, security, consistency. In the previous section we highlighted each of these as a potential pitfall with commercial themes. Interestingly, it's quite the opposite with free themes on *wordpress.org*.

You know that a theme there meets the required guidelines and will be portable. It's been reviewed by the dedicated theme review team so will likely be very secure. All themes are required to use the Customizer for settings, so users can be comfortable in the knowledge that switching themes won't mean learning a whole new options interface. But wait, there's more.

Underscores-based

The majority of new themes submitted to *wordpress.org* are based on Underscores (*underscores.me*), the popular developer-focused starter theme released and maintained by Automattic. Underscores is a rock-solid theme foundation. It means that as well as being familiar from a usability perspective, developers will likely find the code familiar as well. This highlights the portability of these themes. For more on Underscores, see the boxout on page 56.

Community-driven

Popular, free, open source themes are often developed socially on GitHub. They have an active community that reports and fixes bugs, and contributes in other ways, such as providing translations and new features. This helps keep them ahead of their closed source, commercial counterparts.

Financially viable

Some authors are finding new ways to build profitable businesses around their products, while still being able to list them on *wordpress.org* for free. For example, WooThemes (*woothemes.com*) offers its WooCommerce theme 'Storefront' for free on *wordpress.org*, then monetises the project through commercial Storefront plugins and child themes.

This allows Storefront to meet all modern theme development best practises and negate the problems we've covered, while still staying profitable. Other companies have found alternative, successful interpretations of this freemium model. ThemeFoundry's 'Make' theme is a good example. A free version is available on *wordpress.org* while it sells a 'Make Plus' upgrade via its website.

Easily updated

Updating a *wordpress.org* theme is as simple as clicking a link in your WordPress dashboard. You will also get convenient notifications when updates are available, making it less likely that you will miss critical releases.

Accessible

Any theme on *wordpress.org* that is tagged 'accessibility ready' has had a specific accessibility review to ensure that it's accessible to folks with physical and visual impairments. This is an often-forgotten feature that is (in my opinion) one of the most important for any website.

YOUR NEXT PROJECT

When you begin work on your next project, I'd recommend looking beyond the bright lights of the commercial marketplaces, to the humble *wordpress.org* theme directory. The catalogue there is becoming really diverse and I bet you'd be able to find a secure, accessible and attractive theme upon which to base your next WordPress project.

ABOUT THE AUTHOR

RACHEL MCCOLLIN

w: *rachelmccollin.com*
t: @rachelmccollin
Job: Web designer and writer
Areas of expertise: WordPress, mobile and RWD

THEMES

WORK WITH WP CHILD THEMES

Working with parent and child themes can help take your theme development to the next level. **Rachel McCollin** explains how

If you're using WordPress to build websites for yourself or your clients using bespoke themes, then being able to harness the power of child themes will take your skills to the next level.

At their most basic, using child themes will make you more efficient. By keeping the code you use for every project in a parent theme, you'll be adhering to DRY principles. Alternatively, you could use an off-the-shelf parent theme such as Twenty Twelve (*wordpress.org/themes/twentytwelve*), or the parent theme from a theme framework.

Taken further, you can use child and parent themes to create networks of sites with a core codebase, build custom themes based on third-party theme frameworks, or you can even create your own advanced parent theme for use as a theme framework. In this article, I'll give a quick overview of the essentials of child themes and show you some more advanced techniques. You'll learn:

- How to use a child theme to adapt a third-party parent theme or theme framework for the needs of your project
- How WordPress prioritises template files in parent and child themes
- How to override parent theme functions in your child theme's functions file
- How to create your own parent theme for use as a framework, incorporating functions, hooks and filters for use in your child themes

Any theme can act as a parent theme. If you're using a third-party theme and want to tweak it for your own project, it's much better practice to use a child theme to do this than to hack the main theme, which exposes you to the risk of losing your changes when you update the theme to future versions.

CREATING A CHILD THEME

To create a child theme, create a new theme. At the beginning of its style sheet, add:

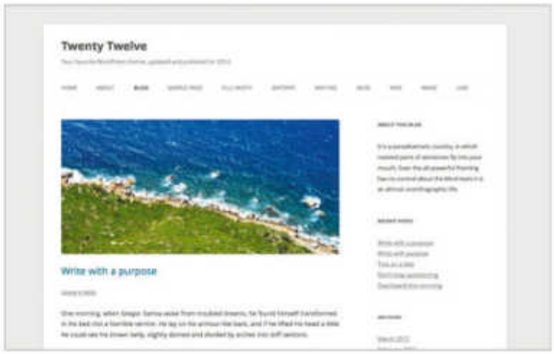
```
/*
Theme Name: My child theme (for example)
Theme URI: URL of the theme or site it's used for
Description: Description of what the theme is for and its main features.
Author: Rachel McCollin
Template: twentytwelve
Version: 1.0
*/
@import url("../twentytwelve/style.css");
```

The important lines are these two:

```
Template: twentytwelve
@import url("../twentytwelve/style.css");
```

The first line specifies the template, which tells WordPress that this is a child theme and that its parent is Twenty Twelve. Note that you use the name of the parent theme's folder here, not the name of the theme (so `twentytwelve`, not `Twenty Twelve`).

The second line imports the stylesheet from the parent theme, so all of the parent theme's styling will be activated in your child theme. You then add your own styling below this `@import` declaration. This means that styling from both themes will be used. However, where a declaration exists in both style sheets for the same element, the child theme's CSS



Twenty twelve This theme is suitable for use as a parent theme

will prevail because of the cascade. A child theme can consist of a style sheet and nothing else if you like – in which case all you'll just use it to override some of the parent theme's styling. Alternatively, you can add extra template files and/or a functions file. You'll then need to understand how WordPress accesses template files from your parent and child themes.

TEMPLATE FILES

The way WordPress uses template files in parent and child themes is very simple. When a given page (or post, or any other content type) is being displayed, WordPress will use the most relevant template file

The functions files in parent and child themes don't interact like style sheets do

from either the parent or child theme according to the template hierarchy (*netm.ag/resources-271*). If it finds two versions of the same template file, it will use the one from the child theme.

This means your child theme's template files will override the parent theme's template files in two scenarios. First, if your child theme contains a template file that's higher up in the hierarchy than those in the parent theme. And second, if your parent and child theme both contain a version of the required template file.

OVERRIDING PARENT FUNCTIONALITY

As well as overriding the CSS and/or template files in a parent theme, you can use a child theme to override the functionality in the parent theme, or add extra functionality. Although if all you're only using the child theme to add extra functionality, you may be better off writing a plugin.

FOCUS ON

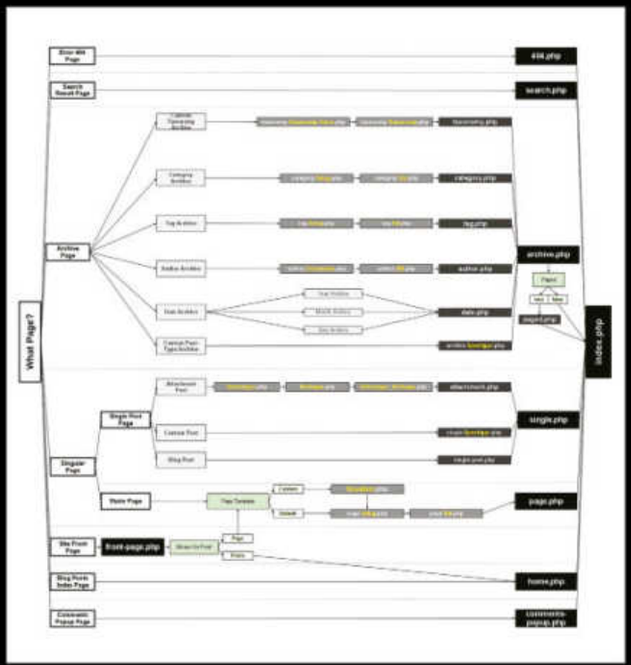
PARENT OR CHILD FILES?

WordPress uses the template hierarchy to determine which template file to use, but this interacts with the files you have in each of your parent and child themes.

Essentially, this takes place through three steps. First, WordPress identifies the type of content being displayed. It then works through the template hierarchy until it finds a file to display that content, either in the parent or child theme. Finally, if the file is present in both the parent and child theme, it uses the one from the child theme.

The table below shows the set of template files in two hypothetical themes, one of which is the child of the other. The files I've highlighted take precedence. Let's look at which template files would be used to display certain types of content.

- **Single posts for the 'product' custom post type:** 'single-product.php', child theme
- **Single posts for other post types (including normal posts):** 'single.php', child theme
- **Listings for the widgets category:** 'category-widgets.php', child theme
- **Other category listings:** 'category.php', parent theme
- **Other archive listings:** 'archive.php', parent theme
- **Search results:** 'search.php', parent theme
- **404 pages:** '404.php', parent theme
- **Pages without a specific template file:** 'index.php', child theme



IN-DEPTH

A SIMPLE FRAMEWORK

The parent theme I use for client site development is a simple theme framework with 12 action hooks, 19 functions and 12 widget areas.

The hooks include `compass_before_header`, `compass_after_header`, `compass_after_nav`, `compass_before_content`, `compass_postmeta_before` and `compass_comments`. I won't bore you with the full list, but needless to say there are action hooks for most areas in the layout. The names correspond to where they appear. Child themes can then use these to insert content, either across a site or in certain locations, using conditional tags.

Functions include:

- `compass_setup`, which sets up the theme
- `compass_before_header_widget_area` (and more functions, one for each widget area)
- `compass_copyright`, which adds a copyright notice, automatically populating the date range
- `compass_comment_display`, which displays comments below any single posts
- `compass_posted_on` adds the date of the post

Widgets are then registered and displayed via the relevant action hooks for the place in the layout where they will appear. In total there are 12 widget areas. Very few sites will use all of them, but different ones will be used for different projects.

The framework uses Object Oriented CSS (*oocss.org*) for layout, meaning that markup in child themes will make use of classes provided by the parent theme instead of adding their own. This means the parent theme's responsive layout isn't overridden by any additional layout classes added in child themes.

The main layout classes are shown in the table below. By using OOCSS, you can make your parent theme responsive as well as avoiding overriding the layout with any layout styling that you then add to your child themes.

Class	Width on small screens (default)	Width on min-width = 321px	Width on min-width = 560px	Width on min-width = 800px	Width on min-width = 1025px
Full-width	100%	100%	100%	100%	100%
Half	100%	100%	100%	50%	50%
One-third	100%	100%	100%	33.3%	33.3%
Two-thirds	100%	100%	100%	66.7%	66.7%
Quarter	100%	100%	50%	25%	25%
Three-quarters	100%	100%	100%	75%	75%
.right	Right-aligned - width dependent on other classes				
.left	Left-aligned - width dependent on other classes				

Unfortunately, the functions files in parent and child themes don't interact like style sheets do. In fact, they work in the opposite way. WordPress calls the functions from your parent theme after those from your child theme, meaning they can override the child theme's functions.

I know this sounds like a bit of a pain; you created a child theme as that's what you want in your site, right? Well, fortunately, there are ways to overcome this. The first method is one you use in your child theme to set the priority when attaching your functions to the relevant hooks. The second is done in the parent theme, and it involves making your functions 'pluggable'.

USING PRIORITY

To activate each function that you add in your child theme, you'll need to attach it to an action hook or filter hook using `add_action()` or `add_filter()`. The `add_action()` and `add_filter()` functions each have three parameters:

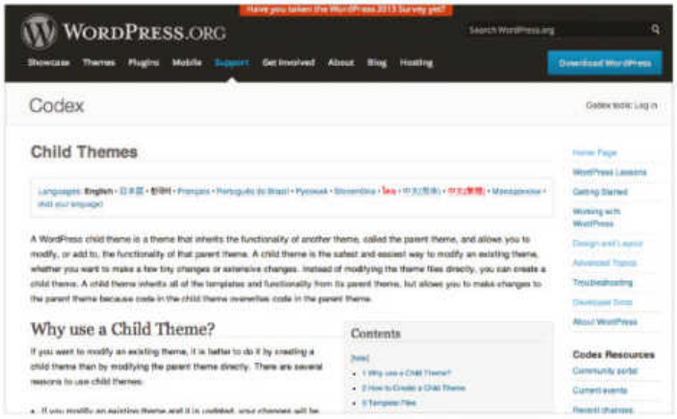
```
<?php add_action( $hook, $function, $priority );
?>
```

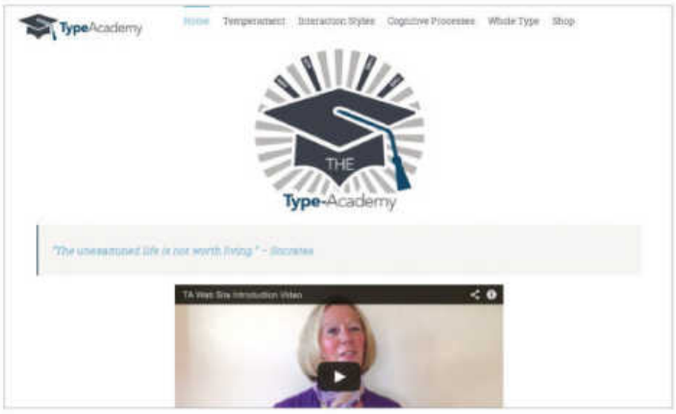
To add your own action hooks to a theme, you use the do_action() function

In most cases, just the first two parameters are used (and are required), but you can use the optional `$priority` parameter to override a function in the parent theme with the function in your child theme. The higher the priority, the later it loads. The default is 10, so if the parent theme doesn't specify a priority, you simply set the priority in your child theme to a number higher than 10.

Let's take a look at how this works. Imagine you're working with a child of the Twenty Twelve theme and you want to override the menu functionality as well as adding your own. This theme includes the `twentytwelve_setup()` function to set the theme up (including registering menus, adding featured image support and more), which is attached to the `after_setup_theme` action hook. The code in Twenty Twelve's functions file is as follows:

```
<?php
function twentytwelve_setup() {
// ... specific functions go here
}
```





Left The Codex page on child themes is a great resource for anyone starting out building child themes

Right Type academy uses a child theme of Compass

```
add_action( 'after_setup_theme', 'twentytwelve_setup'
);
?>
```

To override this in your child theme, you would write a function to replace the one provided by Twenty Twelve and attach it to the `after_setup_theme` action hook, specifying a priority higher than 10:

```
<?php
function my_custom_setup() {
// ... custom functionality goes here
}
add_action( 'after_setup_theme', 'my_custom_setup', 11
);
?>
```

When WordPress encounters these functions attached to the same hook, it will fire the lower priority one first (the one from the parent theme). It will then fire the higher priority one from your child theme, meaning that it can override the function from the parent theme.

PLUGGABLE FUNCTIONS

As I mentioned, there is another method, which you can use if you're creating your own parent themes, and that's to make your functions pluggable.

As WordPress passes the function in the parent theme after those in your child theme, you can code your parent theme's functions so they check for a function with the same name in the child theme. If one exists, the parent theme function isn't passed. You simply use a conditional statement:

```
<?php
if( ! function_exists( 'my_pluggable_function' ) {
        function my_pluggable_function() {
            // function content goes here
        }
}
?>
```

If no function with the same name has already been passed (either in the child or, theoretically, the parent), the function will be passed. But if WordPress has already encountered a function with this name, it will ignore the pluggable function. To make this work, you simply create functions in your child theme with the same name as those in your parent theme, which you wish to override.

Note that this technique will only work if you are creating your own parent theme. Don't be tempted to edit the functions file in a third-party parent theme to make them pluggable. After all, the whole point of creating a child theme is that you don't touch the parent.

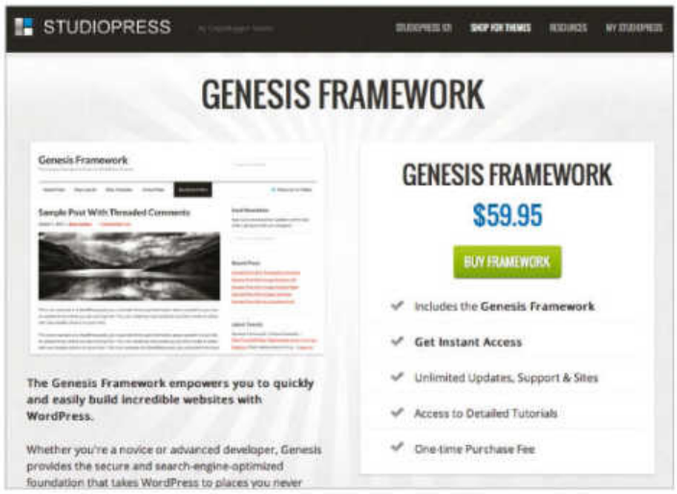
ADVANCED PARENT THEMES

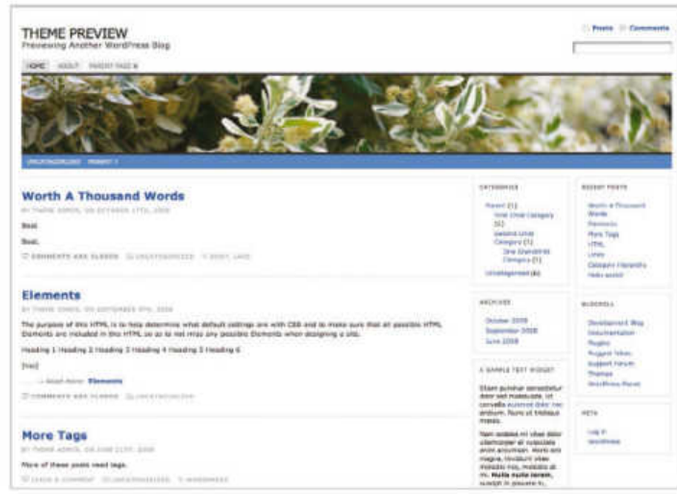
So far I've focused on building child themes, which is the starting point for any WP developer working with parent and child themes. But you can do much more if you build your own parent themes.

For client projects I use a parent theme called Compass, which acts as a simple theme framework. It includes basic template files and minimal styling as well as a CSS reset. It also uses Object Oriented CSS (OOCSS) for layout. I can incorporate responsiveness into the parent theme without that being overridden by any layout styling added to child themes.

It includes a number of pluggable functions that can be overridden in child themes if needs be. It also includes hooks, which don't necessarily have any functionality attached to them in the framework but can be used to add content or functionality in child themes.

For example, the theme has 12 action hooks relating to specific locations in the page layout, meaning they can be used to add content in those locations. So, the `compass_footer` hook can be used to add static content, ▶





Left The Genesis theme framework uses a parent theme with multiple hooks

Right Free theme Atahualpa is used as a parent theme for customisation via a child theme

loops or widget areas in the site footer. In the parent theme, it's used to place widget areas (so long as these are populated). In child themes it can be used for much more.

To add your own action hooks, use the `do_action()` function, which takes the following parameters:

```
<?php
do_action( $tag, $arg1, $arg2, $arg3 etc....);
?>
```

The `$tag` parameter is the unique name of the action hook, which you then use when attaching a function to that hook. You can add as many arguments (`($args)`) as you need. These are more useful for plugin development than for creating parent themes. So, in Compass, there's a hook called `after_nav()`, which is used to insert content immediately after the navigation menu:

```
<?php
do_action( 'compass_after_nav' );
?>
```

To add content in that location in a child theme (without having to edit the template files), I then attach a function to it using `add_action()`:

```
<?php function my_example_function() {
/// function goes here
}
add_action( 'compass_after_nav' , my_example_function'
);
?>
```

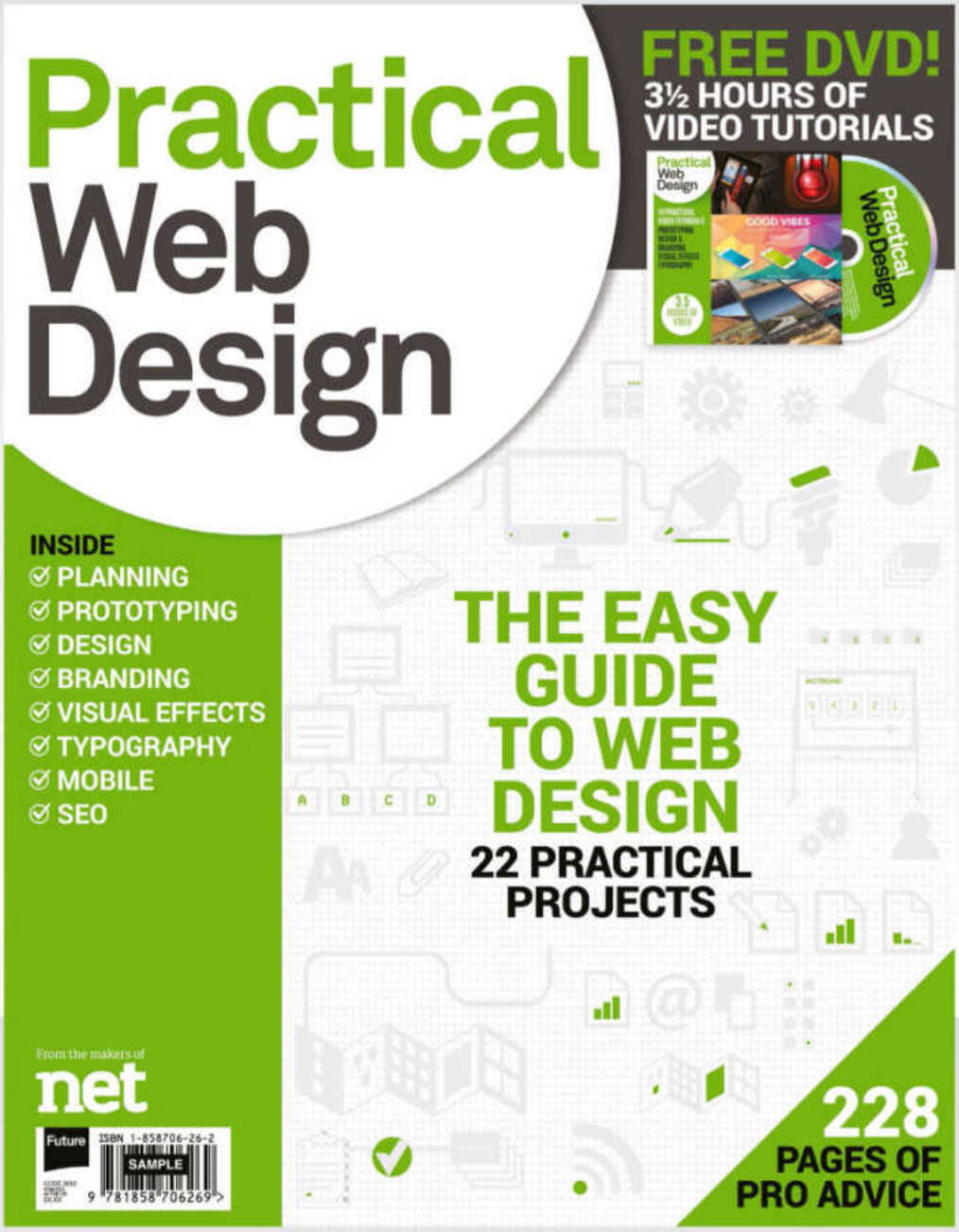
You can also add your own filter hooks, which work slightly differently from action hooks. Instead of being used to insert content, you use them to change the way content or data is output – for example, to change the way metadata such as the date of a post is output.

By combining these techniques and using them throughout your parent theme, you can create a framework to act as a starting point for all of your projects using child themes. Example uses include:

- Inserting content in specific places where action hooks have been created, such as the header, footer, sidebar or before or after the content
- Adding action hooks in a template file (such as 'front-page.php') to enable insertion of content via the child theme's functions file
- Creating custom functions you can use in your child themes. For example, in Compass I have a function called `compass_is_child()` that checks if the current page is the child of a specified page
- Creating filters allowing you to output data differently in your child themes to the parent theme

The big theme frameworks make extensive use of hooks. The Genesis Framework has over 50 hooks and the Thesis framework has over 40. By adding hooks to your own parent theme you can create a framework that will make your child themes more efficient and powerful, with more flexibility in terms of what you can create for your and your clients' development projects.

Developing using child themes can boost your efficiency and effectiveness as a WordPress developer. Take this further by using the child theme's template files and functions file to override or supplement functionality from the parent theme. Also try building your own parent theme, giving you a quick starting point for all new projects and allowing you to add a lot more to your child themes with less effort than is required when starting from scratch.



This practical guide to web design shows you how to build better sites the easy way. We'll guide you though the process from start to finish – from prototyping approaches to mobile techniques and SEO. Plus, the FREE DVD comes packed with 3.5 hours of video tutorials to help you master top tools including Photoshop, Sketch and Pixate, in no time.

Sharpen your skills now!

Available at all good newsagents, and online at
netm.ag/practicalwebdesign-268

View source files here!

All the files you need for this tutorial can be found at *netm.ag/genesis-268*

ABOUT THE AUTHOR

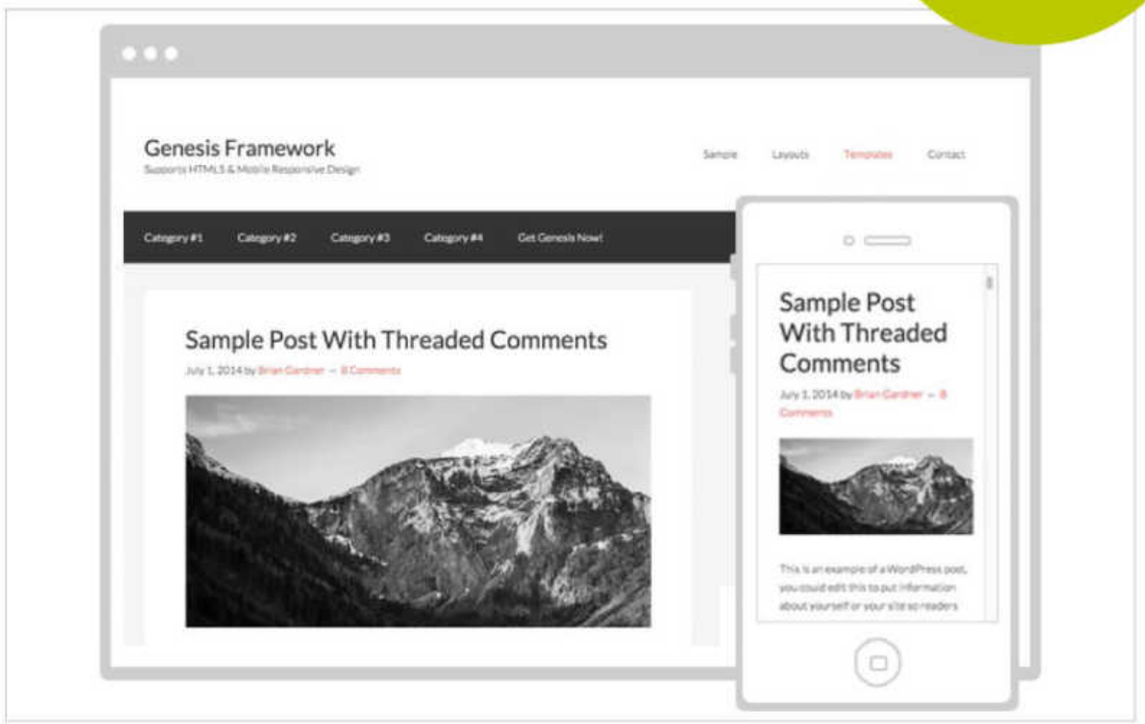
CARRIE DILS

w: *carriedils.com*

t: @cdils

Job: WordPress developer and consultant

Areas of expertise: WordPress, Genesis Framework, PHP, CSS, HTML



GENESIS

GET STARTED WITH THE GENESIS FRAMEWORK

Carrie Dils introduces the Genesis Framework and explains how to get started working with child themes

In the WordPress world, frameworks tend to refer to some sort of starter or boilerplate code. By starting new theme build projects with a framework, you save development time, reduce repetitive tasks, and generally give yourself a head-start on projects. While all of that is true for Genesis, in this case the word 'framework' can be a little misleading.

The work of StudioPress, the Genesis Framework (*netm.ag/genesisWP-268*) is simply a parent theme, albeit a pretty pimped out one. It's a toolbox for building child themes, which gives you a consistent output of HTML markup and a basic CSS style sheet to use as a starting point for your customisations. It's easy to assume such a robust theme will be bloated or full of unnecessary code, but Genesis is surprisingly lightweight (the entire zip is just over 400KB). It's coded (and documented) very well, extremely secure, and integrates *schema.org* with its HTML5 markup, giving users an SEO advantage right out of the gate.

Genesis includes pre-built components for all of your standard web layouts – navigation menus, headers, sidebars, footers and so on. It's like a set of Lego – its modular structure enables you to add, move or remove anything.

BUILDING BLOCKS

Here's where Genesis differs from other parent themes, instead of relying heavily on templates,

VIDEO

Carrie Dils has created an exclusive screencast to go with this tutorial. Watch along at *netm.ag/genesisvid-268*

Building block Working with Genesis is like working with Lego – you can tear it apart and rebuild it any way you want (Image: Curtis McHale)

child theming with Genesis is primarily about working with action hooks and filters. Genesis provides a base from which all of your Lego blocks can be shifted, replaced or modified, without ever touching a template file (although you certainly can use templates). The most basic child theme only requires a style sheet and a 'functions.php' file.

In short, Genesis is incredibly flexible and poses no limitations on development. If it can be done with WordPress, it can be done in Genesis.

Child theming is primarily about working with action hooks and filters

FILE STRUCTURE

At some point, even if you don't call yourself a developer, you'll want to customise your theme in some way. When the moment comes for you to drop in a code snippet or write your own function, your work will go more smoothly if you have a conceptual overview of the framework.

If you unzip Genesis and look at the file structure, you'll see the root files are really no different from what you'd expect to find in any WordPress theme: there are a dozen or so standard template files and a style sheet. From there, various folders contain files for specific bits of functionality. The magic file is in the '/lib/' folder: 'framework.php':

```
function genesis() {
get_header();
do_action( 'genesis_before_content_sidebar_wrap' );
```

IN-DEPTH

IS GENESIS RIGHT FOR YOU?

Still undecided about if Genesis is for you? Let's take a look at the pros and cons for different types of users ...

DESIGNER

Genesis will not limit your creativity – designing child themes for Genesis is no different than designing for any other WordPress theme. If you're not a techie, don't worry, there are a ton of developers whose business is built around taking your Photoshop or Illustrator files and converting them into themes.

DEVELOPER

Instead of starting from scratch with each theme build, the Genesis Framework provides the core functionality and files to get you off the ground running. Of course, you can get as complex as you want, extending the WordPress API and core Genesis functions to suit your requirements. There's not one thing in Genesis that can't be added, moved, removed, filtered or changed.

DIY USER

The Genesis Framework boasts an engaged community that is continually churning out tutorials, which makes it great for DIY-ers. That said, Genesis doesn't include any fancy site builder tools and changing fonts isn't as simple as checking a box. While there are some premium plugins that make customisations simpler, Genesis isn't as easy to work with as some other theme frameworks.

CONSULTANT

Genesis is a great solution if you provide client services that include websites. By using Genesis, you create a consistent starting point for all of your web projects, making your workflow more efficient. Additionally, the framework has such wide use in the WordPress world that, if needed, you could find freelance designers or developers to help with a project.

Wide audience Even though Genesis is built with developers in mind, it appeals to a broad audience, including designers, agencies and DIYers

FOCUS ON

HOW TO PICK THE RIGHT CHILD THEME

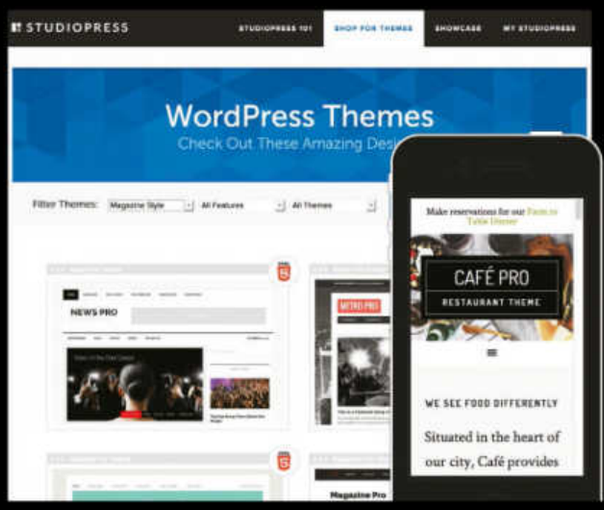
Finding the perfect theme to showcase your content is a bit daunting. I suggest starting at *StudioPress.com*, the company that created Genesis. With over 50 child themes, it's the largest repository of themes for Genesis on the web. If you can't find what you want there, go to Google. There are a ton of quality third-party theme sellers, myself included. Here are some tips for narrowing down the best theme for you:

1 Don't waste time on themes that aren't mobile responsive. It's a baseline standard to have a site that looks good on mobile devices. If a theme isn't advertised as mobile responsive, move on.

2 Go for HTML5 markup. The Genesis Framework offers support for HTML5 markup and integration with *schema.org*. Even if you don't know how to leverage this immediately, do yourself a favour and start off on the right foot.

3 Beware free themes. While you can find some wonderful free themes for the Genesis Framework around the web, these generally don't come with any customer support. If you anticipate needing help further down the line, look for themes that include support with the purchase.

The bottom line? Using the criteria above, find the theme that gets you 90 per cent of what you're looking for. You can always customise that last 10 per cent. If, after this process, you still can't find a theme, consider looking for (or building) a custom theme.

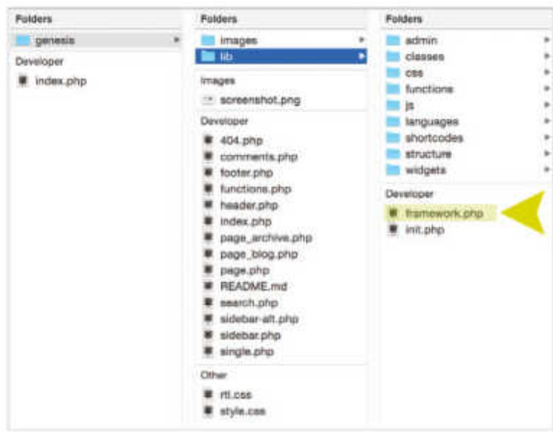


Theme store *StudioPress.com* has the largest theme store for the Genesis Framework, and its filtered search makes it easy to quickly browse themes

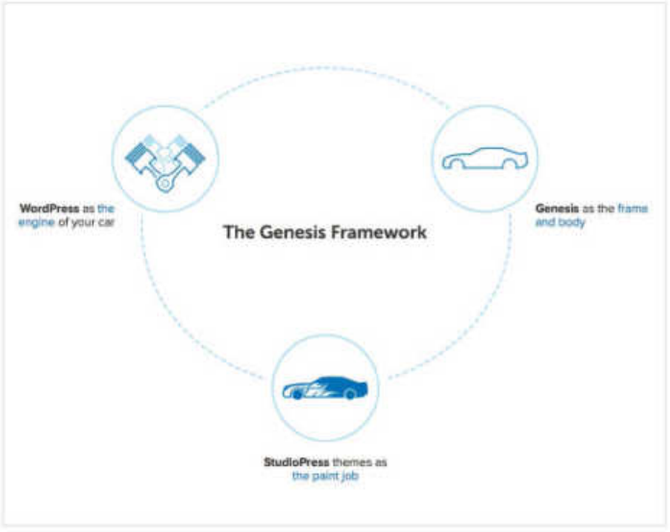
```
genesis_markup( array(
'html5' => '<div %s>',
'xhtml' => '<div id="content-sidebar-wrap">',
'context' => 'content-sidebar-wrap',
) );
do_action( 'genesis_before_content' );
genesis_markup( array(
'html5' => '<main %s>',
'xhtml' => '<div id="content" class="hfeed">',
'context' => 'content',
) );
do_action( 'genesis_before_loop' );
do_action( 'genesis_loop' );
do_action( 'genesis_after_loop' );
genesis_markup( array(
'html5' => '</main>', //* end .content
'xhtml' => '</div>', //* end #content
) );
do_action( 'genesis_after_content' );
echo '</div>'; //* end .content-sidebar-wrap or #content-sidebar-wrap
do_action( 'genesis_after_content_sidebar_wrap' );
get_footer();
}
```

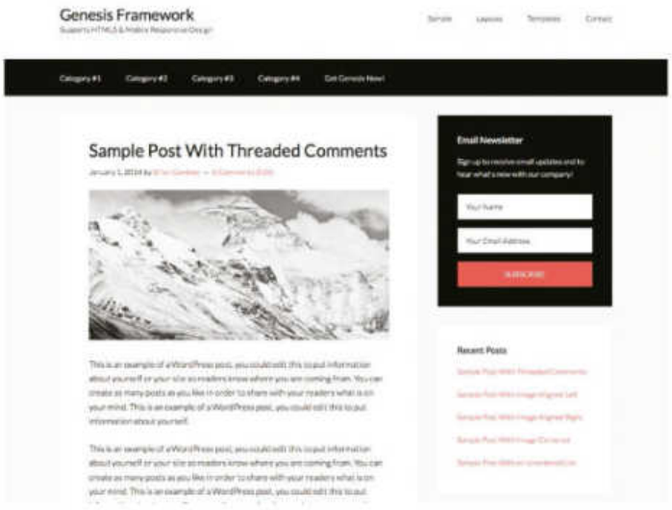
This file is the heart of the Genesis Framework and is really the only part that can't be changed. It's the foundation upon which everything else is built. The `genesis()` function declared here is called from nearly every standard template file. When this function is called, 'framework.php' is loaded and all the code within this function is processed.

If you've done any theming or customising in WordPress, the `get_header()` and `get_footer()` functions probably look familiar. Those call in the 'header.php' and 'footer.php' files respectively. In-between those functions is the basic structure



Functions file 'Functions.php' is the heart of the Genesis Framework. When the genesis() function is called in a file, this page loads





Left What is the Genesis Framework? Think of WordPress as the engine, Genesis as the body, and child themes as the paint

Right The Genesis Sample theme is a bare-bones example of a child theme, and is a great starting point for experimentation

of every page, including the Loop (which displays the current post or page), some basic HTML markup, and seven action hooks. If you're curious and want to follow the rabbit hole, do a universal search in the framework files for each of those hooks and you can see what actions (or even additional hooks) are attached to them.

A child theme is a way to remove the things you don't want and add in the things you do

All in all, there are over 50 hooks in the Genesis Framework that enable you to inject custom code almost anywhere on the page. StudioPress has put together detailed documentation on each of these hooks, along with examples of use, at *netm.ag/hooks-267* (available to members only).

MAKING CUSTOMISATIONS

Just like with the WordPress core, you should never directly edit files from the Genesis Framework. Always make your customisations via a child theme (we'll talk more about this in a bit). If you make edits to the actual framework, they'll be lost the next time you apply an update. If you stick to working with a child theme, your customisations are safe from being overwritten and you can update Genesis anytime an update is available.

If you're 'tinkering' with code customisations, you'll do the vast majority of your work in the child theme's 'functions.php' file. From there you can universally apply code (e.g. remove post meta throughout the whole site), or combine your functions with WordPress conditional statements to target specific content (e.g. remove post meta on any posts in the 'Comedy' category). You can also overwrite any default Genesis functionality by creating specific template files. For example, if you have a custom post type for 'Movies', you can use the standard WordPress template hierarchy to create an archive or single template file to display your movies.

My rule of thumb for whether to use 'functions.php' or a custom template file is how many customisations I'll make. For instance, if I can write a handful of functions to do what I need, I'll just add those to 'functions.php'. If it's more complex than that, I'll consider using a custom template.

For style changes, I recommend going directly to 'style.css'. Since you never need to update child themes (that's the point of using them), you can directly edit the style sheet or any other file and make it yours. Really, there's no right or wrong way to add your customisations. It's more about learning the most efficient way to get the results you want, and following good coding principles as you go.

CHILD THEMES

So far we've talked about the basics of Genesis and customisations. Now let's look at child themes. First off, Genesis child themes cannot be used on their own. The Genesis Framework must also be installed (but not activated) on your WordPress site. From there, you can activate any Genesis child theme. As with the traditional WordPress parent/child theme structure, a Genesis child theme automatically inherits everything from the parent framework. You can think of a child theme as a way to remove the things you don't want from Genesis ▶

Get creative Head to *studiopress.com/showcase* to see how other designers have customised Genesis child themes

and add in any extra bells and whistles that you do. Also, the child theme is where all of your custom styles are. Genesis itself includes very little CSS, as it's not intended for use as a standalone theme.

If you want as basic a theme as possible, start with the Genesis Sample theme, available for free on GitHub (or in your StudioPress account, if you have one). The theme is visually very sparse, so it's a good place to begin making both style changes and code changes. I've found the best way to learn is to open up the 'functions.php' of your child theme and start experimenting with code. My favourite starting point is to look at 'post.php' in the Genesis source code and pick any add action from the `genesis_reset_loops()` – the function that spits out the guts of a post, from the image at the top to the comments section at the bottom.

Since we already talked about post meta, I'll copy that `add_action()` statement and paste it into 'functions.php':

```
add_action( 'genesis_entry_footer', 'genesis_post_meta' );
```

Next, change the `add` to `remove`. This reverses, or overrides what Genesis would output on its own.

```
remove_action( 'genesis_entry_footer', 'genesis_post_meta' );
```

That's a simple example and probably not practical (it removes the post meta everywhere on your site, including your blog page, archive pages and single posts), but it demonstrates how you can work with Genesis via your child theme.

Don't be timid about trying things out in your development environment. You may white-screen it a bit, but experimentation really is the best way to get comfortable with the framework (or any code, for that matter). I also recommend using Google liberally. There are hundreds, if not thousands, of Genesis tutorials out there. There are even formal courses available at both Lynda.com and Treehouse.

IN CONCLUSION

If you're new to theme development, you'll find Genesis is very approachable, especially considering the bounty of free learning resources online. If you've already spent some time in WordPress theme development, you'll have a definite learning curve with Genesis, but once you're over the hump, you'll love what you can do with it.

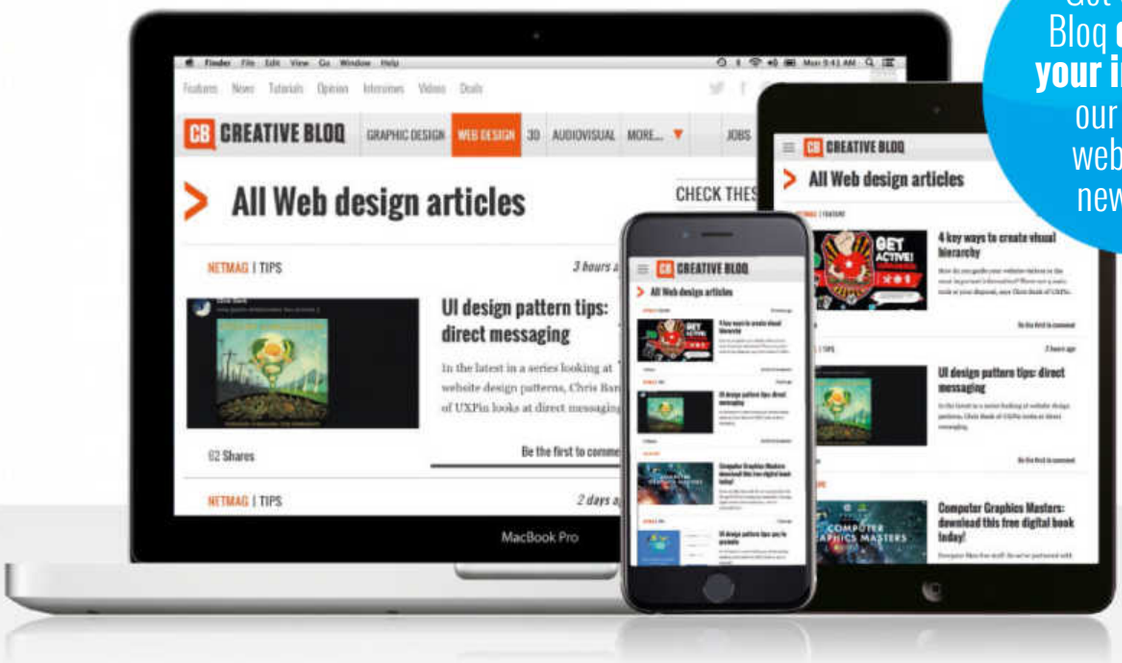
Used by over 130,000 folks and under active development, Genesis isn't going anywhere any time soon. If you plan to stick around in the WordPress theme industry, it's definitely worth a look. Who knows, it might become your favourite tool for building websites.

TOP TIP

If you're looking through Genesis source code, you'll notice support for both XHTML and HTML5. The HTML5 markup was added in the Genesis 2.0 release, but the XHTML code remains for backwards compatibility.

Themes (L–R): Bill Erickson, Northbound Design, Jared Atchison

The number one destination for **web design** news, views and how-tos.



Get Creative Bloq **direct to your inbox** with our weekly web design newsletter

CB CREATIVE BLOQ

Graphic design | **Web design** | 3D | Digital art

www.creativebloq.com

View source files here!

All the files you need for this tutorial can be found at *netm.ag/childgit-271*

ABOUT THE AUTHOR

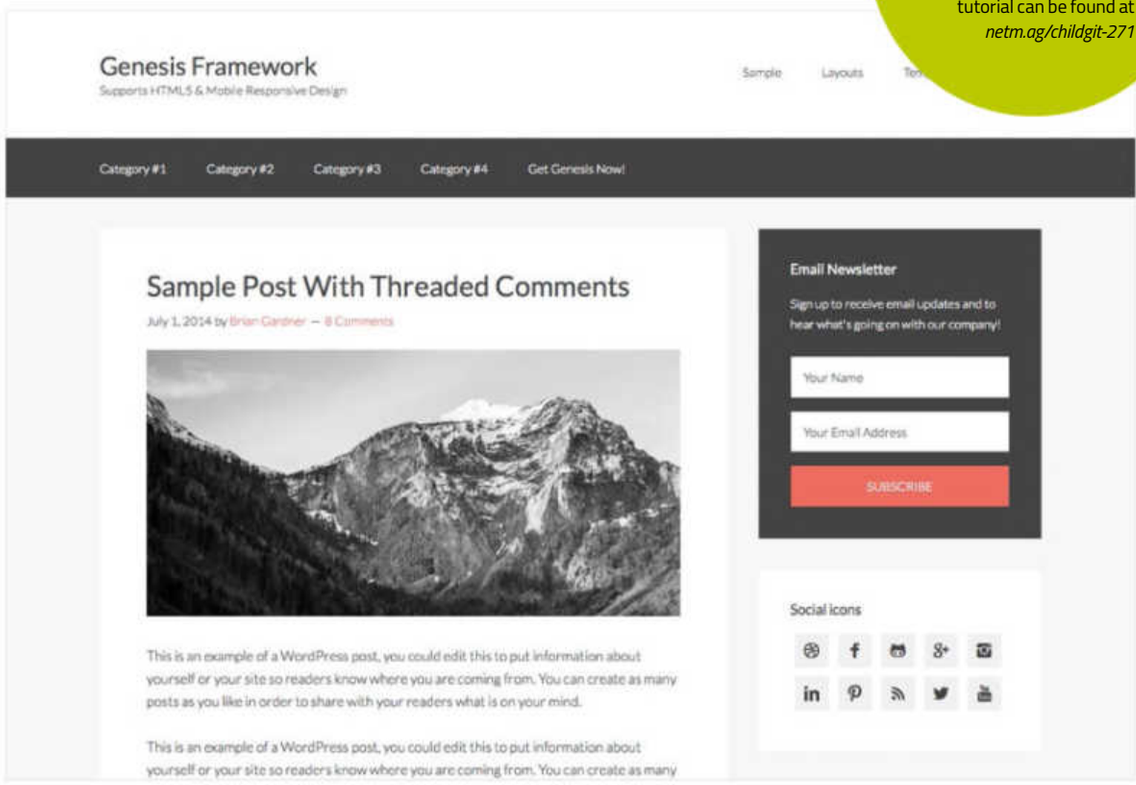
CARRIE DILS

w: *carriedils.com*

t: @cdils

Job: WordPress developer and consultant

Areas of expertise: WordPress, Genesis Framework, PHP, CSS, HTML



GENESIS

BUILD YOUR OWN CHILD THEME WITH GENESIS

Carrie Dils explains how you can take a child theme and customise it – or even create your own from scratch

In this tutorial, I'm going to take a look at building themes using the Genesis Framework. Let's start by taking a moment to get on the same page regarding what a WordPress theme framework is and where the Genesis Framework falls into that definition. The WordPress Codex (basically the Bible of WordPress: *codex.wordpress.org*), defines theme frameworks in three ways:

1. A `drop-in` code library used to facilitate the development of a theme
2. A stand-alone base/starter theme that is intended to be forked into another theme
3. A parent theme template

The Genesis Framework (*netm.ag/genesisWP-268*) falls into the third category – it is meant to serve as a parent theme. In practical terms, this means if you have a WordPress site with the Genesis Framework installed, any Genesis child theme you activate will inherit all of the functionality of the framework (in other words, the parent theme).

The benefit of this structure is that you never need to directly edit files in the Genesis Framework, which leaves it safe to update. Instead, you make customisations via a child theme, which you should never update. To summarise, in the WordPress world, if you want to change the parent theme, you need to edit the child.

RESOURCE

Custom child themes generally include at least one custom featured image size. This plugin lets you regenerate thumbnails for any existing media images: *netm.ag/regenerate-271*

So, how does this structure translate to the Genesis Framework where, by default, you're already working with a child theme? Do you then change the child theme by creating and editing a grandchild? No. While technically possible, you'll be hard-pressed to find support for a grandchild theme. As an alternative solution, I recommend treating the child theme as unique and making your customisations directly.

MAKING A THEME YOUR OWN

If you're new to child theming, I recommend starting out with an existing Genesis child theme and editing it to make it your own. What does that mean? Let's take a look at a practical example.

I'm starting a new web project for a real estate company called Major Realty. There are a few existing Genesis child themes for real estate – I'll choose the Winning Agent Pro theme as a starting point. The design of the theme is similar to what I want my finished theme to look like, so it makes sense to use it as a base.

Before I make any customisations, there are four things I need to do to make this theme unique. Let's take a look at these now.

If you're new to child theming, start out with an existing theme and make it your own

1 Rename and move the theme folder

When you buy a theme, it's typically available for download as a zip file from StudioPress or whatever third-party vendor you purchased it from. I like to store this zip in an 'original copies' folder along with other fresh downloads.

From there, I'll unzip the theme (leaving the original zip as it is) and rename the unzipped folder to match my new theme name. In this example, I'll rename the 'winning-agent-pro' folder 'major-realty'. Then I'll move the theme folder over to my local WordPress install's themes folder ('wp-content/themes/') and place the 'major-realty' folder under version control.

2 Change the theme name in style.css

If I open up the 'style.css' file in the 'major-realty' theme folder, there's a documentation block (or 'doc block') at the top of the file that shows the theme name along with some other details.

FOCUS ON

BLANK SLATE VS STARTER CHILD THEME

Ask 100 developers how they prefer to create a theme and you'll get 100 different answers. The benefit of starting with a blank file in a code editor is that you don't carry over unnecessary code. You only create the code you need. The benefit of starting with a starter child theme is that you can re-use code you've already written, speeding up your development cycle.

So, which is better? Well, there are 100 different answers to that too, but here's the general guideline I recommend:

If you're new to child theming

Don't start from scratch. Find a Genesis child theme that most closely resembles the end result you want and customise it to match exactly what you need. Over time you'll develop a level of comfort working with theme files and a basic understanding of working with the unique hooks and filters in Genesis.

If you're happy modifying existing themes

Start from scratch – at least once. It doesn't have to be complex or even for a real project, but building a theme from the ground up will help cement your knowledge. As you add only the code you need to get the result you want, you'll gain a far deeper understanding of the Genesis Framework.

If you're really finding your groove

Create your own starter theme. There's no need to re-invent the wheel every time you build a new theme – as a developer, you'll have elements you always like to incorporate. By including these in your starter theme, you'll save yourself time with each new project. Also, use version control, as your starter theme will constantly evolve.

FOCUS ON

RESOURCES

These are my go-to tools and resources for every theme I build. Also noted are some articles related to child theming to help you get started on your first theme.

Articles

5 things I do when building a custom Genesis theme (*netm.ag/gardner-271*) – Nobody knows child theming better than Brian Gardner, the creator of StudioPress and the Genesis Framework. Learn more about his process by reading this article.

Plugins

Genesis Visual Hook Guide (*netm.ag/hook-271*) – While you're getting used to the hooks and filters available in Genesis, I highly recommend using the Genesis Visual Hook Guide plugin, freely available in the WordPress plugin repository. Use it on any page of your site to see exactly which hooks and filters are in use and where they occur. I use it for just about every site build.

Query Monitor (*wordpress.org/plugins/query-monitor*) – This is another indispensable tool. Run it on any page to see which template file is in use, which WordPress conditional statements apply, every hook fired (in order), and more.

References and tools

PHP Documentation Standards (*netm.ag/standards-271*) – Another handy resource is the PHP Documentation Standards guide, written by the core WordPress team. I refer to it frequently to make sure my code documentation is in line with what other WordPress developers are doing.

CSS Lint (*csslint.net*) – To help you write the cleanest CSS possible, use CSS Lint to check your code for errors, compatibility, performance and more. It's also available to use on command line.

Develop sites locally

If you're not developing WordPress sites locally yet, now's the time to start! Whether you're on a Mac or a PC and want to install the components manually or use a 'do it all for you' tool, here are some resources to get you started.

How to Install WordPress Locally for PC/Windows with XAMPP by Raelene Wilson (*netm.ag/local-271*)

Configuring a Local Apache/PHP/MySQL Dev Environment in OS X by Brian Richards (*zen.net/local-develoment-in-osx/*)

Get DesktopServer and save time! (*netm.ag/time-271*)

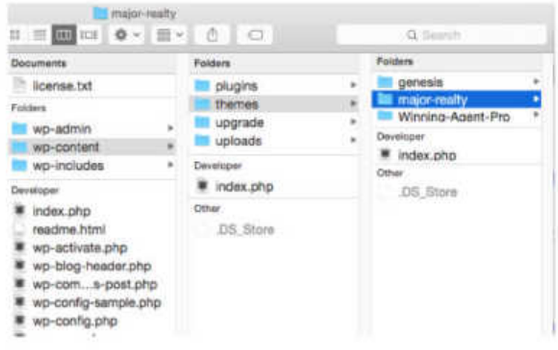
```
<pre>
/*
    Theme Name: Winning Agent Pro Theme
    Theme URI: http://www.winningagent.com/go/winning-agent-pro-theme/
    Description: A mobile responsive and HTML5 theme built for the Genesis Framework.
    Author: Carrie Dils
    Author URI: http://www.carriedils.com/

    Version: 1.0.0

    Tags: black, gold, red, blue, green, orange, white, one-column, two-columns, fixed-width, custom-menu, full-width-template, sticky-post, theme-options, threaded-comments, translation-ready
```

Updating the theme's screenshot makes it easier to identify in the WordPress admin

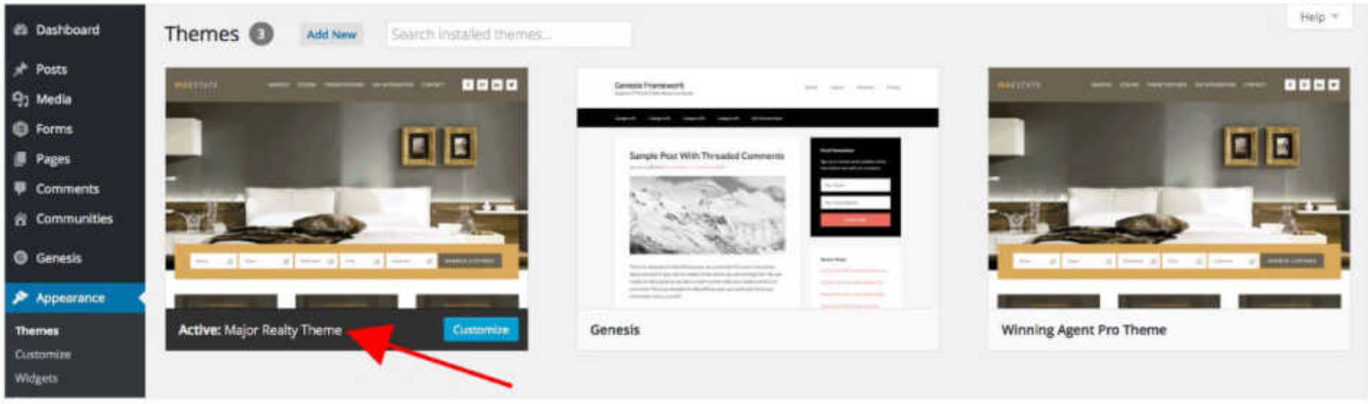
```
    Template: genesis
    Template Version: 2.0

    License: GPL-2.0+
    License URI: http://www.opensource.org/licenses/gpl-license.php
*/
</pre>
```

I want to change the theme name on line two from `Winning Agent Pro Theme` to `Major Realty Theme`:

```
<pre>
/*
    Theme Name: Major Realty Theme
</pre>
```



Change of name Renaming the folder means I'll never mistakenly apply an update for Winning Agent Pro to Major Realty.



WordPress admin
You can see the theme name change from 'style.css' appear in the WordPress admin under the Appearances > Themes menu

By doing this, I've indicated in the documentation that this is no longer the Winning Agent Pro theme. I've also made it easier to identify the Major Realty theme from my WordPress admin.

3 Update the doc block in style.css

While I'm in 'style.css', I'll update some other items in the doc block. There are a couple of reasons why this is a good idea. A year from now, if I want to go back and make a change to the Major Realty theme, this documentation will help me remember certain details, such as which theme I used as a base.

Also, should another developer eventually work on this theme, they'll appreciate knowing these details. It's especially useful to know when a theme is based on another.

```
<pre>
/*
    Theme Name: Major Realty Theme
    Theme URI: http://www.winningagent.com/go/winning-agent-pro-theme/
    Description: A mobile responsive and HTML5 theme built for the Genesis Framework. Based off the Winning Agent Pro Theme
    Author: Carrie Dils, Sherry Mills
    Author URI: http://www.cdils.me/

    Version: 1.0.0

    Tags: black, gold, red, blue, green, orange, white, one-column, two-columns, fixed-width, custom-menu, full-width-template, sticky-post, theme-options, threaded-comments, translation-ready

    Template: genesis
    Template Version: 2.0

    License: GPL-2.0+
    License URI: http://www.opensource.org/licenses/gpl-license.php
*/
</pre>
```

I've changed the theme name, but left the theme URI as is, pointing to the original Winning Agent Pro documentation. That documentation might come in handy at some point, so it's good to leave that reference intact. I've updated the description to indicate that Major Realty is based on the Winning Agent Pro theme.

In this case, I'm the original author (spoiler alert), but I'd recommend adding your name – either in conjunction with the original author or as a replacement – to indicate that you've done some work on the theme. It's also worth updating the Author URI to reflect your domain (again, it's very helpful to know where to find the original author if another developer ends up working on the theme). The rest I'll leave as it is.

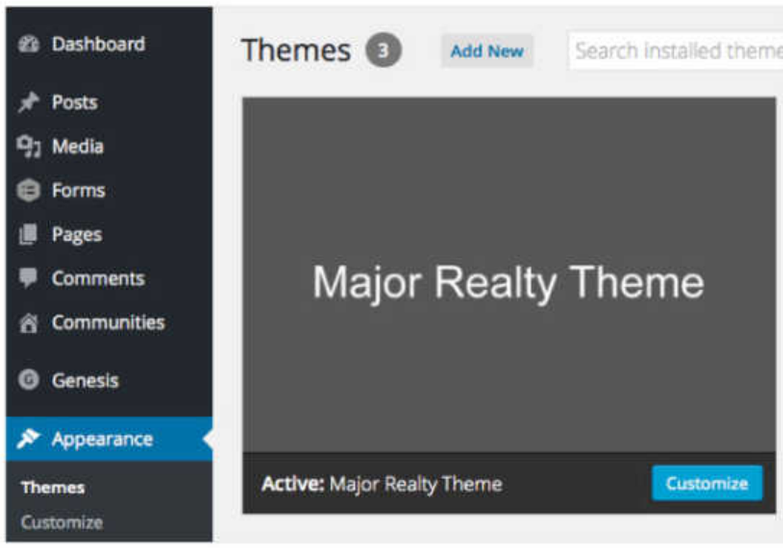
There's one important thing to note before we move on: because this theme is originally licensed under GPL, anything I make based on it (i.e. my own child theme) must keep the same licensing.

4 Change the screenshot

While not technically necessary, updating the theme's screenshot makes it easier to visually identify (and differentiate) the theme in the WordPress admin. The screenshot doesn't need to be fancy.

The easiest method I've found for doing this is to open up the theme folder, find 'screenshot.png' and then open it in an image editor. This ensures that I'm starting with the right image dimensions. For simplicity, I've given it a solid background colour, overlaid text with the theme name, and saved it back to its original location.

Once I've taken these four steps, even without ▶



Spot the difference If I had Winning Agent Pro and Major Realty installed, I could easily tell them apart

changing a single thing about the appearance of the theme, I've made it a unique child theme. Now I'm ready to begin customisations.

USING HOOKS AND FILTERS

The Genesis Framework includes numerous action hooks and filters you can use to either modify default behaviours or add new functionality. This means that child theming with Genesis is more about utilising hooks and filters than creating templates, as you would with a more traditional WordPress theme. For example, let's say you were creating a child theme based on a theme you've downloaded from *wordpress.org*. If you wanted to change the footer, you'd create a new template file for your child theme named 'footer.php'. This would override the 'footer.php' file from the parent theme. Conversely, with Genesis you'd use one of the many available hooks and filters to change the footer – no new template needed.

If you've done a lot of child theming before, doing things the 'Genesis way' requires a significant mental shift. Once you get used to it, though, you'll love how easily you can make changes.

PRO TIPS

Here are some lessons on child theming I've learned along the way. These tips will save you time and leave you with leaner code.

Develop locally

If you're developing on a live site, you're wasting time. Skip the constant FTPing files and waiting on browser reloads by running WordPress locally.

RESOURCE

It's important to know whether functionality belongs in your theme or should go in a plugin instead. In this article, I explore just that. *netm.ag/dils-271*

There are a couple of requirements you need to do this: an Apache server running PHP and a MySQL database. You can set up those components yourself or use a tool like DesktopServer (*netm.ag/desktop-271*) to do it for you.

Ditch the extra CSS

Unless you're starting with a blank 'style.css' file, your stylesheet will include things you don't need. For instance, you can get rid of theme colour options, since you don't need them for a custom build. Other things you might remove are elements you know you won't use, like certain layouts or widget areas.

Keep your stylesheet tidy

Make sure the table of contents in your 'style.css' file stays up to date with the items you've added or removed. Staying organised makes your life (at least the coding part of it) easier. When you add new styles, don't stick them all at the bottom of the stylesheet – add them in context. For example, if you're adding styles for a plugin, group it with the other plugin styles.

Skip the constant FTPing files and waiting on reloads by running WP locally

Document your code

Document every bit of code you write. If you copy and paste code from a tutorial or another site, document that too. When you're in the build process, all of your code is familiar and it's tempting to skip the documentation, but when you step away – even for a few days – you'll forget what a bit of code was for and waste your time reorienting yourself. Give your future self the gift of documented code!

TO SUM IT ALL UP

The art of child theming really is an art. While there's certainly some science – such as correctly written code – the process of building a child theme is a little different for everyone. When you're starting out, it's helpful to see how other developers structure their workflow when working with themes, but with each new theme you build, you'll define your own workflow and find the things that work best for you.

The most important thing to remember? Have fun coding and don't be afraid to break things! That's the best way to learn.

This all-new guide includes everything you need to do more with JavaScript, the programming language of the web. Through 27 practical projects, you'll learn how to speed up both your workflow and the performance of your sites and apps.

Become a JavaScript master today!

Available at all good newsagents and online at
netm.ag/javascript-265

View source files here!

All the files you need for this tutorial can be found at *netm.ag/RWD-271*

ABOUT THE AUTHOR

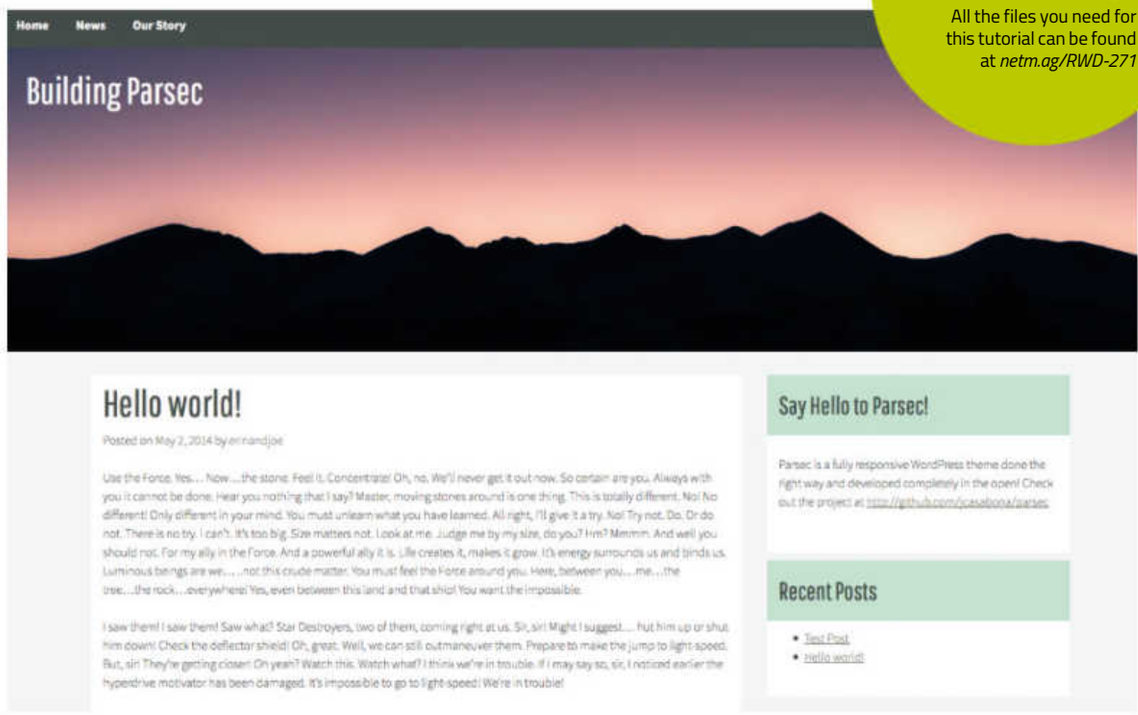
JOE CASABONA

w: *casabona.org*

t: @jcasabona

job: Frontend developer

areas of expertise: HTML, Sass/CSS, PHP, WordPress



RESPONSIVE

CREATE A RESPONSIVE WORDPRESS THEME

Joe Casabona walks through the tools and processes he is using to build his reponsive Parsec theme

In 1977, one of the greatest movies of all time came out: *Star Wars: Episode IV – A New Hope*. Most consider it George Lucas' masterpiece. He put a lot of time and effort into creating this movie. He introduced revolutionary special effects and much more, making it a timeless classic.

In 2005, *Star Wars: Episode III: Revenge of the Sith* came out. This is a bad movie. Lucas rushed it, not as much time and effort went into it, and it was shot primarily in front of a green screen. Lucas got lazy and everyone knew it. It doesn't hold a candle to that first movie.

Why am I telling you this? Because there's an important lesson here: if you get lazy, people will know. We also know it's easy to get lazy or feel rushed. 'This has a really tight deadline' sound familiar to anyone else? Last year I gave a talk preaching best practice for responsive design, but I wasn't doing those things myself. I decided to start practising what I preached.

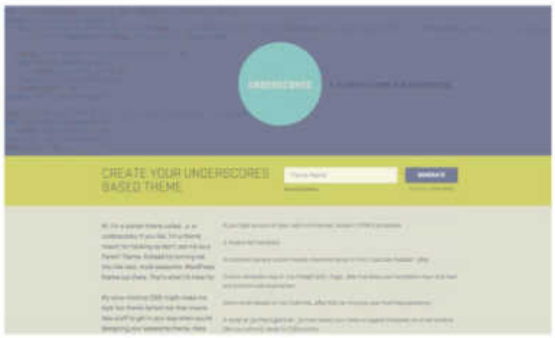
THE GOAL

Since my book *Responsive Design with WordPress* came out in late 2013, a lot has changed. I started using Git on a daily basis. I started using Sass and got introduced to automated tools like gulp. Browsers have advanced quite a bit. For my new template, Parsec (*github.com/jcasabona/parsec*), it was time to update my tools and process (see boxout opposite).

From a personal standpoint, I wanted to improve my process and learn some new tricks. From a deliverable standpoint, I wanted to create a 'good' responsive theme. That means I needed to make a theme that:

RESOURCE

This article by Luke Wroblewski explains why good UIs are obvious, and hiding elements can confuse and put off users: *netm.ag/wrobleski-271*



Underscores I cannot say enough great things about this starter theme. It's perfect for WordPress theme development

- Is mobile-first
- Has content-based breakpoints, not device-specific ones
- Is lightweight and loads quickly

THE TOOLS

First, let's look at the tools. These will change over time. When I started writing my book, I didn't use Sass, and MAMP was my local development environment. I didn't use GitHub for anything besides finished code – it would have been great for my Millennium Flights theme (*netm.ag/flights-271*) and the accompanying plugins.

Using GitHub means readers have the chance to see the development process. It makes it possible to create something, change it and try something new – then to merge the changed version into the finalised implementation. It can also help you write cleaner code. With that in mind, I knew GitHub would be instrumental for Parsec. Plus, by using the wiki and issues features, all the documentation could be made public.

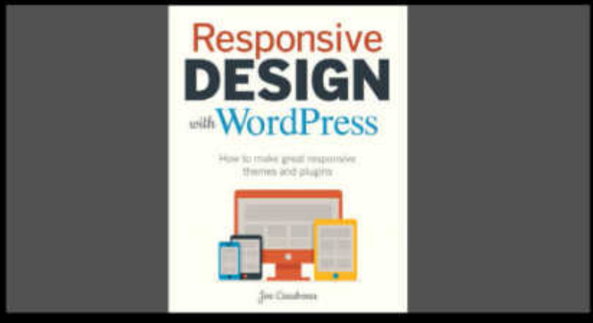
Starter theme Underscores makes just the right number of decisions for you

When developing the Parsec theme, I started with the incredible starter theme Underscores, or _s if you prefer (*underscores.me*). I did this for a few reasons. Underscores has a strong community backing. It's developed by the same folks who develop WordPress' default themes, and it also has about 90 contributors.

Underscores includes code for some common WordPress customisations and well-organised CSS. It makes just the right number of decisions for you. ▶

IN-DEPTH

THE PROJECT



RWD I started writing *Responsive Design with WordPress* over two years ago. While still a good resource, a lot has changed since then

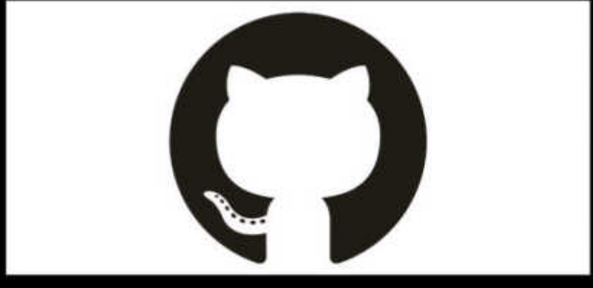
I felt the best way to accomplish the goal I set for myself (namely to improve my process and tackle responsive design for WordPress the right way), was to create a brand new project and document it from planning to launch. The project I decided on is an open source theme called Parsec, along with some related plugins, both custom and downloaded.

I also decided I would do everything from sketches and notes to code and issue tracking, out in the open. GitHub is essential in achieving this. I have my main project branch – the master – as well as a couple of feature branches I'm working on.

On top of that, there are two completely separate project branches that are based on Parsec. I'm adding in certain features that may get merged, but for the most part, these are standalone projects that treat Parsec as a framework.

Within the master branch there are two folders: 'wp-content', and 'assets'. 'wp-content' is what you might expect: the theme and plugins. 'assets' is a folder with any raw assets Parsec may make use of. At the time of the writing, there are Style Tiles and possible fonts.

Finally, GitHub's wiki and issues areas are also put to good use. The issues help me track new features, to-dos and bugs. The wiki includes any resources I have used and may use for reference.



In the open GitHub is a great tool for project management in general, and integral to keeping this project out in the open

It supports enough for two types of layouts and a mobile menu, providing users with a good starting point without having to undo anything.

Finally, Underscores supports Sass, another tool I use heavily. Sass is an incredible piece of software. It allows you to build extensible, flexible CSS. Since I'm using Parsec as a starting point for my themes, my code is organised in such a way that I should be able to swap colours and fonts easily. Sass helps with that.

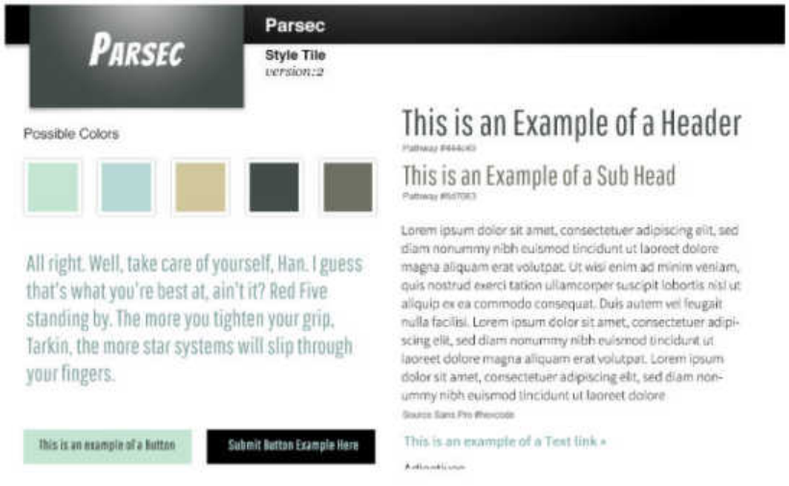
THE PROCESS

Whenever I design a website, I follow the same steps. Let's take a look at those now:

0 Choose the content
1 Sketch for narrow and wide widths
2 Create a Style Tile
3 Code
4 Test

In true programmer fashion, I started this list with a zero. My justification is that content should be ready before the design process starts. This is something I tell everyone. It's also something I do for all my projects. I know expecting this for every project sounds like an 'in an ideal world' situation, but it's so important!

I explain the importance of starting with content by asking a simple question: 'How can I know what is most important if the content isn't finished?' At the start of the process, knowing how the content stacks is integral. This is especially true when designing mobile-first. Not only does it indicate to me the purpose of the website and what is most important to users, it also prevents me from adding extra elements. I'm looking at you, mega-menus, parallax and image carousels. Once the content is ready, we start the process.



The dark side This is version two of my Style Tile for Parsec. In my final version, I made the colours lighter and added some cleaner fonts

SKETCH

To me, sketching is a must-have part of the design process. This stage allows me to lay out the content and see how things might look before putting any code on the page.

It also forces me to consider some important questions. Parsec is intended to be a basic, almost starter theme, so early on I needed to figure out what exactly I want to use it for. Would it be primarily for personal sites? Business sites? Blogs, wedding sites or event sites? As I started to sketch, I realised I didn't know the answer to this question.

Through sketching, I was able to illicit some answers. I'm making the theme for myself first, so what sites do I need to make? My current personal site uses WordPress' default theme, so that could use a facelift. My wedding is coming up. That will also need a website.

In the end, I decided Parsec would be a personal site theme. That said, if you look at the GitHub repo, you will see I was able to create a wedding site and an events site using the same base theme.

Sketching is the freest form of design in this medium. It means you can start with a blank slate, not confined by the tools you're using – whether they be code, Photoshop, Sketch or some other programme. You aren't limited by the assumptions you'd make in front of a computer screen.

STYLE TILES

One drawback of sketching is that you can't get a good idea of how elements will mesh together. For this, determining fonts and font sizes, colours and other graphics is necessary. That's where Style Tiles come in.

A big focus of my design process, and something I've been encouraging others to do for a while is to avoid using Photoshop. When you're designing websites with responsive, fluid layouts, using a set Photoshop canvas can be detrimental. How do you decide what widths to design at? Are you going to make three or four different mockups for the different widths, showing how – but not when – the layouts should change? In my opinion, going from sketch to code or some other live layout updating program like Macaw is a better option.

But still, you may want to pair colours, fonts and graphics to see if they play well with each other. Style Tiles (*styletil.es*) is a nice substitution for Photoshop, and allows you to communicate a design's visual brand.

I use Style Tiles to play with colour schemes, choose font faces and sizes, and see how certain graphics look. I am able to iterate a lot quicker when I don't have to worry if certain font pairings will

work, or how well the colours I chose play together. It's thanks to Style Tiles that I ended up with the nice light and clean colour scheme on Parsec instead of the darker version I came up with first (both Style Tiles are in the GitHub repo).

Once the Style Tile is complete, I use it as a reference for the design, first setting up my Sass variables. This is another fantastic advantage to using both Style Tiles and some kind of CSS preprocessor – the Style Tile translates nicely into Sass variables.

START CODING

With sketches and styles in hand, it's time to put code to editor. Since Underscores gives us a bit of a head start, the first thing I do is define Sass variables and see what the page looks like out of the box.

First, I define some base colours, then create element-specific variables based on those colours:

```
$color-primary: #333333; //dark grey
$color-secondary: #C05300; //burnt orange
$color-accent-two: #006776; //teal
...
$color-background-body: $color-white;
$color-background-screen: lighten($color-gray-base, 36%);
```

Sketching means you can start with a blank slate, not confined by the tools you're using

After that, I assign some variables for font families and sizes, breakpoints, padding and more. This is all part of the 'structure' of the site, helping keep consistent vertical rhythm and improving the design. Adding breakpoints as variables is especially helpful. You can quickly change or add more based on your content.

The last piece I do for setup is some common mixins. I won't share all them here, but I do want to point out two helpful ones:

```
@mixin breakpoint($breakpoint, $direction: min) {
        @media (#{$direction}-width: $breakpoint)  {
                @content;
        }
}

@mixin breakpoint-tweener( $bp-start, $bp-stop ) {
        @media screen and (min-width: $bp-start) and
(max-width: $bp-stop) {
                @content;
        }
}
```

Engagement levels Luke Wroblewski notes that after Polar's navigation changed to a less obvious design, engagement plummeted

The first is your standard breakpoint mixin. You pass an em value and either `min` or `max`, and the media query gets built for you.

The second is what I call a 'breakpoint tweener' – or styles that belong only within two specific breakpoints. It's not too often I use this, but often enough to warrant a mixin for it.

IMPLEMENTING NAVIGATION

Earlier I mentioned that Underscores has a nice way of implementing responsive navigation: you tap a 'menu' button to reveal the hidden navigation. I changed this to a hamburger icon, and I will likely optimise it when I implement gulp.

One note: earlier this year, Luke Wroblewski wrote a great article (*netm.ag/wrobleski-271*) about how obvious is better when it comes to UI. The gist is that hiding elements with an icon might look nicer, but it negatively affects engagement. The word 'menu' for example, is more obvious than the hamburger icon, even if the latter seems ubiquitous.

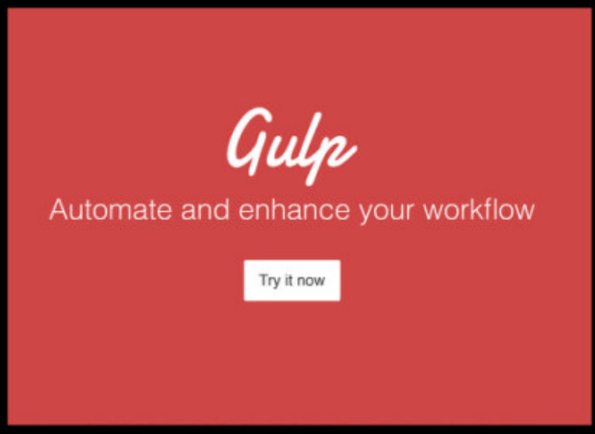
IMPLEMENTING IMAGES

There are several ways to do responsive images. I'm using a double-pronged approach: multiple versions of images in CSS, and the `<picture>` element inline. First the CSS:

```
.impact-img {
  background: url( $asset-path + 'img/impact-sm.jpg' ) center
no-repeat;

  @include breakpoint( $breakpoint-1 ) {
    background-image: url( $asset-path + 'img/impact-med.
jpg' );
  }
```

FOCUS ON

FUTURE WORK



Asset size gulp will serve as a great tool to help reduce the size of assets, from JavaScript to images

There are a few things that, at the time of this writing, had not been completed with Parsec. I didn't include them in the main body text because I wanted to speak about my experience so far, but here's what I have planned for the future.

Something I plan to implement is gulp. gulp is one of many JavaScript-based automation tools that can perform tons of great tasks – including basic ones like compiling Sass, and combining and minifying JavaScript. It can also perform powerful tasks like creating SVG sprites and icon fonts on the fly, or connecting with Git for deploying code. This tool in particular will go along way in optimising Parsec's code and reducing load times.

In line with a point that Luke Wroblewski made about obvious design being better, I plan to change the hamburger icon currently serving as the menu button back to the word 'menu' in a future iteration. This may allow me to eliminate font icons completely in the base design, while also providing a better user experience.

On the server side, I want to implement a solid RESS (REsponsive design with Server Side detection) function, making use of `wp_is_mobile()` as well as some sort of API, like Scientiamobile. This will allow me to dynamically determine a device and serve up assets accordingly. This means I will only need to print one type of navigation – or perhaps I will be able to exclude some JavaScript fallback because I know the user's device supports a feature natively. The possibilities are endless. In the end, we should have a rock-solid responsive WordPress theme!

```
  @include breakpoint( $breakpoint-3 ) {
    background-image: url( $asset-path + 'img/impact.jpg' );
  }
}
```

As you'll see, I use smaller images to start, and as the screen gets bigger, I load in bigger images. While not a perfect assumption, I decided that if you're on a smaller screen, you are likely on a mobile device. In this case, I use an image that will load faster and doesn't use as much data. The same sort of logic is used for the `<picture>` element.

A great piece of news came out around June of 2014: the `<picture>` element had become an official part of the HTML spec, meaning we could start using it! This element allows us to define more than one image source, then define breakpoints to display each image:

```
<picture>
  <source srcset="extralarge.jpg" media="(min-width: 1000px)">
  <source srcset="large.jpg" media="(min-width: 800px)">
  <img srcset="medium.jpg" alt="This is the alt tag, and default image">
</picture>
```

For RWD, there are three major testing areas: browser, device and connectivity

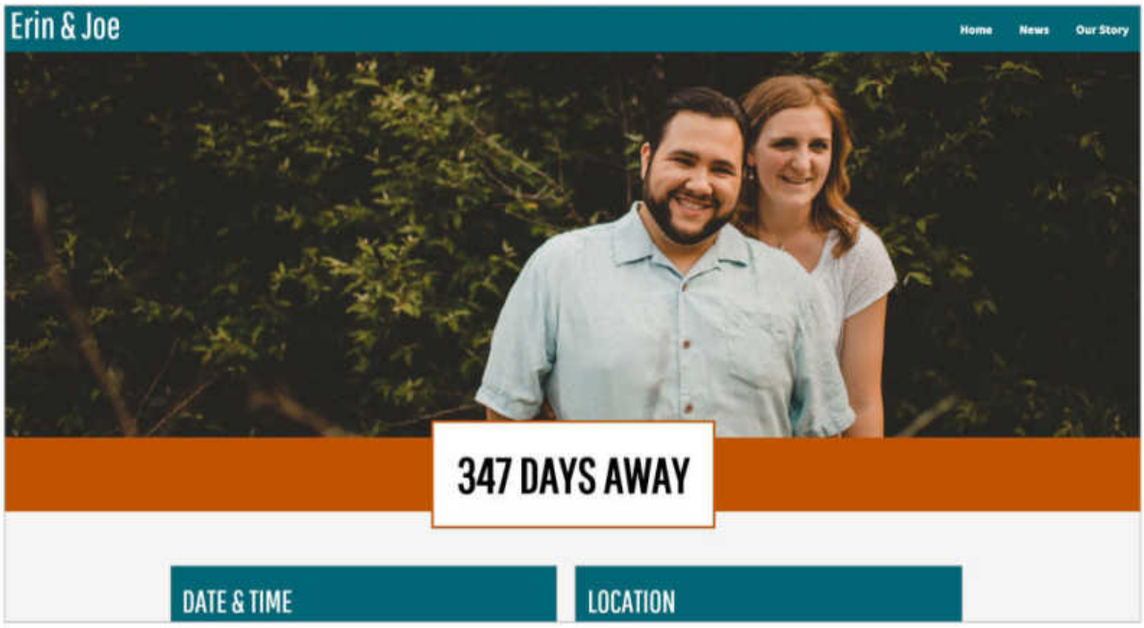
The only problem is that it's not supported by all browsers yet. Luckily there is picturefill.js and a fantastic plugin called RICG Responsive Images (*netm.ag/plugin-271*) to help with this. After writing my own implementation for responsive images, I decided this plugin was worth a look.

At the time of this writing, I am still testing it. You can see my implementation in the wedding branch of the project.

TESTING

I want to close out this article with some tools I'm using for testing. I want to make it super-clear that no matter what, you should test on actual devices. First, let's talk about how you should test. For responsive design, there are three major areas of testing: browser, device and connectivity.

Browsers are what we're most familiar with, and responsive design didn't make things any easier for us. On top of desktop browsers, we now need to test



Parsec fork My wedding website is a fork of Parsec that includes some nice features and styles, proving Parsec's flexible base structure

mobile OSs and the browsers they use. This non-comprehensive list includes:

- Safari on iOS
- Chrome on iOS
- Firefox on iOS
- Chrome on Android
- Browser on Android
- Firefox on Android
- IE for Windows Phone

Of course, the browsers you choose to support depends on your user base. In the case of Parsec, the more support, the better.

Then there is devices. Part of the reason for the content- versus device-based breakpoints debate is the sheer number of different screen resolutions around (over 6,000 on Android alone, last time I checked). Different mobile OSs also handle certain things differently.

You need to determine which devices you should test, but I usually recommend:

- The latest and previous versions of the iPhone
- The iPhone 6+
- The latest and previous versions of Google's Nexus device
- An Android 4.0 device
- The iPad and iPad Mini
- A 7" Android Tablet
- A 10" Android Tablet
- A Windows Phone

This can be tough, because you may not have these devices at your disposal. There are a few things you can do. I use BrowserStack, which works well, but nothing substitutes actual devices. Adobe EdgeInspect gets you part of the way there, as it will allow you to simultaneously test any devices you own at the same time.

You can try find a device lab (*opendevicelab.com*) or ask some friends if you can test on their devices. If worst comes to the worst, you could always walk into your local AT&T store and load your site on the devices there. Efficient testing and free advertising!

Finally, you should test connectivity and loading times. Try testing on Wi-Fi, 4G and 3G at a minimum. Slower if you can. Do this while moving and standing still – you will get a much better idea for how fast your site loads, allowing you to adjust accordingly. Remember, 80 per cent of users will abandon a site if it takes more than five seconds to load.

FINISHING THE PROJECT

As I write this, I'm still working on Parsec. I hope to have it ready to go sometime in late 2015, with support for gulp and RESS. You can watch my progress here: *github.com/jcasabona/parsec*.

I had lofty goals and an aggressive timeline for this project. But I also wanted to put the proper effort into it, test it as much as I could, and do things the right way. I want this to be an *Episode IV*, not an *Episode III*.

Download the files here!

All the files you need for this tutorial can be found at *netm.ag/wordpress-254*

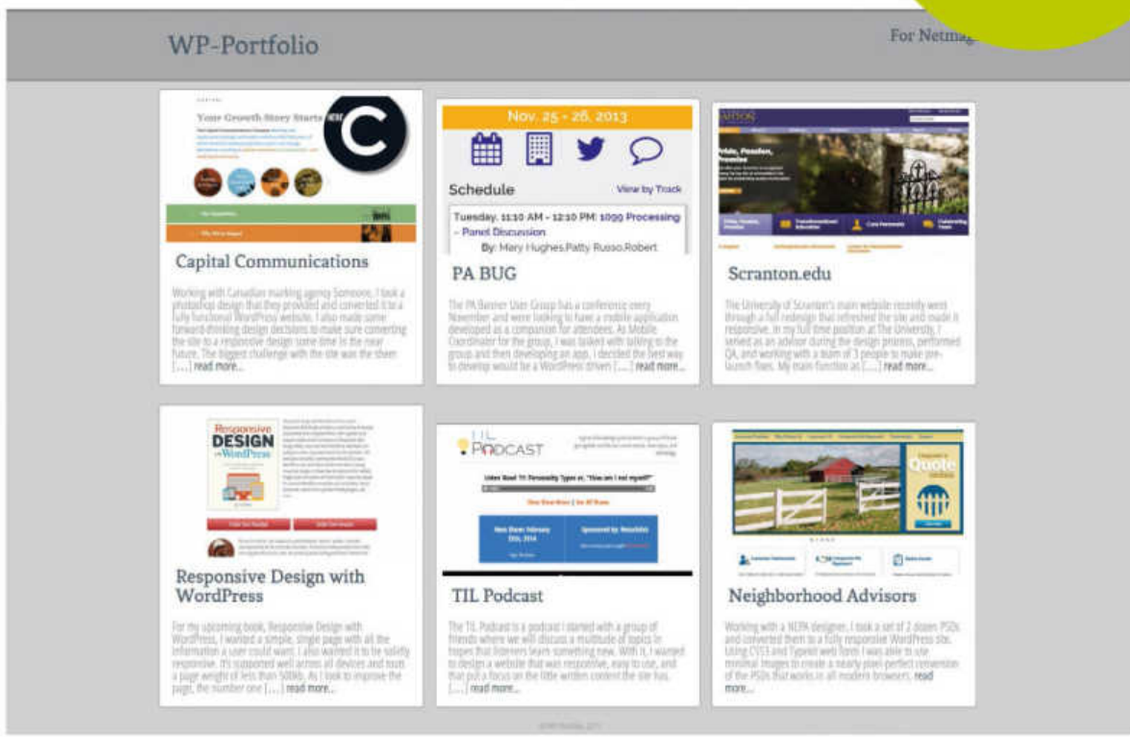
ABOUT THE AUTHOR

JOE CASABONA

w: *casabona.org*

t: @jcasabona

areas of expertise:
HTML, CSS, PHP, MySQL, WordPress



RWD

BUILD A RESPONSIVE WORDPRESS PORTFOLIO

Joe Casabona explains how to create a plugin to set up a new Custom Post Type and a new theme template, for your own responsive portfolio

Web development may change rapidly, but two things that are here to stay are WordPress and responsive design. Knowing how to build responsive WordPress themes and plugins is a must. In this tutorial, we will look at building a WordPress plugin and theme template for a responsive portfolio.

We will work with two mocked-up templates: an archive page, which will list all of the recent projects, and a single page, which will show a specific project. The archive page for the portfolio is a pretty simple one with a header and three columns of projects at full width. This will shrink to two columns, then one column, as the screen gets smaller. The HTML and CSS is available at GitHub: *netm.ag/wordpress-254*. Each project on the page will have this structure.

This is the HTML that will be generated by the WordPress Loop:

```
<div class="card">
    <img src="project.png">
        <h3>Name of Site</h3>
        <p>Short description and a link <a href="#">read more...</a></p>
</div>
```

The single page is going to have a similar layout, wrapping all of the text in a container called `.project` instead of `.card`. The CSS is also going to be fairly lightweight and scaled. You may notice in the site files there is a `style.scss` file. I've developed all

VIDEO

Joe Casabona has created an exclusive screencast to go with this tutorial. Watch along at: *netm.ag/video-rwd-254*

the CSS using Sass, but fear not: the generated `style.css` is compiled in a readable way.

CREATING A NEW CUSTOM POST TYPE

A Custom Post Type is an object in WordPress that allows us to add any types of content we want to the WordPress editor, treating them the same way as posts. In a fresh WordPress install, there are menu options for Posts and Pages, each of which handles content differently. With Custom Post Types, we can add options for creating new types of content to the WordPress admin menu. We'll create a Custom Post Type called Portfolio.

We're going to develop this Custom Post Type as part of a bigger WP plugin for portfolio projects. While we could add the functionality to the theme, this is bad practice because then our content is tied to our design: if we change the theme, we lose the portfolio. We will handle display through two

RWD is here to stay. Knowing how to build responsive WordPress plugins is a must

methods: templates/template tags, and shortcodes that can be used through the editor.

The first step is to define the plugin. Create a folder in '/wp-content/plugins/' and name it whatever you like. I've named mine '/jlc-projects/'. Inside that folder, create a file of the same name (for example, 'jlc-projects.php') and add this code:

```
<?php
/*
Plugin Name: Joe's Portfolio Plugin
Plugin URI: https://github.com/jcasabona/wp-portfolio
Description: A simple plugin that creates and display a
projects portfolio with WordPress using custom post types!
Author: Joe Casabona
Version: 1.0
Author URI: http://www.casabona.org
*/
define('JLC_PATH', WP_PLUGIN_URL . '/' . plugin_basename(
dirname(__FILE__) ) . '/' );
define('JLC_NAME', "Joe's Portfolio Plugin");
require_once('jlc-project-cpt.php');
?>
```

There are a few things going on here. The first is the standard plugin definition for a WordPress plugin; the next few lines create constants and then include the ▶

FOCUS ON

LEARNING SASS

Sass (*sass-lang.com*), which stands for Syntactically Awesome Style Sheets, is a CSS preprocessor that lets you write CSS in a way similar to programming languages. You can define variables, make comments hidden from the output file, write functions and extensible CSS classes, perform mathematical operations, and more.

As someone who primarily considers himself a programmer, I find this method of writing very appealing. I can abstract away a lot of what I would normally repeat. Plus, as far as RWD goes, Sass makes it much easier to keep all of the media queries for one class together. For example:

```
.card{
        /* Default Styles Here */
        @media screen and (min-width: $bp-medium){
                display: inline-block;
                width: 40%;
                }
        @media screen and (min-width: $bp-large){
                width: 44%;
                }
        }
```

No matter what size your project is, I would recommend using Sass for the CSS. It will make managing your style sheets a lot easier, and make responsive sites better, through improved organisation and by compiling multiple resources into one.

If you want to learn Sass, the best resource out there is *Sass for Web Designers* by Dan Cederholm (*netm.ag/Sass-250*).

Book apart Dan Cederholm's *Sass for Web Designers* is an excellent Sass primer

FOCUS ON

RWD AND WORDPRESS

This article only scratches the surface of responsive design using WordPress. There are a lot of other factors to keep in mind. Remember that developing responsive websites is about more than just the site shrinking on small screens. You should also ensure that you're using best practices by:

- **Using ems for breakpoints**

These are much more accessible for users who may have different browser settings and handle the wide variety of screen resolutions much better.

- **Using content-based breakpoints**

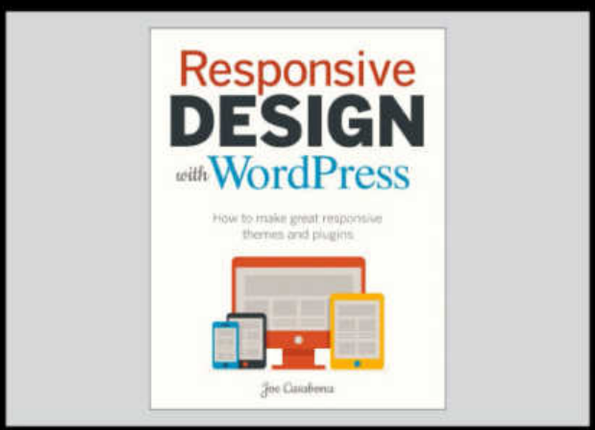
Make sure you're developing in such a way that your breakpoints are determined by the content of your website and not the device's resolution. That way you know your content looks good universally, not just on three specific devices.

- **Keeping speed and size in mind**

In a perfect world, everyone would be on Wi-fi or 4G and not have to worry about data caps. Unfortunately, that isn't the case. As we develop websites, we need to remember to load only what's needed, reduce page weight and remove extra HTTP requests. This will make things much easier on the user.

Tools like Sass and WordPress can help us achieve these goals, especially the final one. In my recent book, *Responsive Design with WordPress* (*rwdwp.com*) I go over best practices for responsive design, why they are important and how to implement them in WordPress, as well as pitfalls to avoid.

I also include several tutorials like this one in the last chapter, taking you through developing forms, photo galleries, and even a products page responsively in WordPress.



Further reading Joe Casabona's book complements and extends this tutorial

files the rest of the plugin needs. At this point, there is only one other file: 'jlc-project-cpt.php'.

You will also notice that I'm using the prefix 'JLC_' (or 'jlc-') for everything. You should choose your own prefix to use. Prefixing variables and function names will decrease the chance of your plugin conflicting with other themes or plugins.

Before we jump into 'jlc-project-cpt.php', I want to add one more bit of code to 'jlc-projects.php'. The code below will create a new image size, which we will use with our Custom Post Type:

```
if ( function_exists( 'add_theme_support' ) ) {
    add_theme_support( 'post-thumbnails' );
    add_image_size('jlc_project', 1100, 640, true);
}
```

Now it's time to create 'jlc-project-cpt.php'. I'll only be discussing the important code here, but you'll find the complete code on the GitHub repo. First (after the opening `<?php` tag) we define the Custom Post Type:

```
add_action('init', 'jlc_projects_register');
function jlc_projects_register() {
    $args = array(
        'label' => __('Portfolio'),
        'singular_label' => __('Project'),
        'public' => true,
        'show_ui' => true,
        'capability_type' => 'post',
        'hierarchical' => true,
        'has_archive' => true,
        'supports' => array('title', 'editor', 'thumbnail') ,
        'rewrite' => array('slug' => 'portfolio', 'with_front'
        => false)
        );
    register_post_type( 'portfolio' , $args );
    register_taxonomy("jlc-project-type", array("portfolio"),
    array("hierarchical" => true, "label" => "Project Type",
    "singular_label" => "Project Type", "rewrite" => true));
}
```

This is your standard Custom Post Types definition function. We add an action to call it on `init`, then send

New UI The admin page to add a new project. Notice the Project Link section

our list of arguments to `register_post_type()`, along with the type's slug, which will be 'portfolio'. After registering the post type, we register the custom taxonomy to go along with it. It's important to keep these two functions together. If you don't, and the taxonomy somehow gets registered first, WordPress will throw an error.

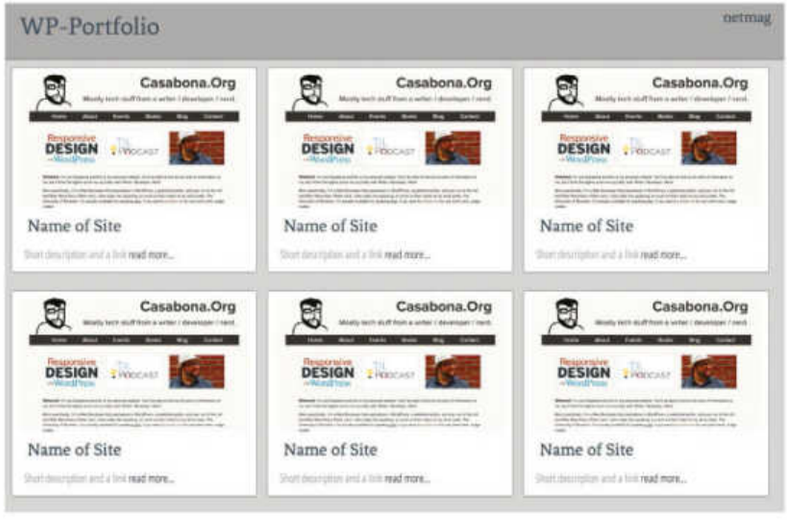
After the Custom Post Type is defined, it's time to add the custom metadata we want to use. Our Custom Post Type supports a title, the editor (which will serve as the body text), and a thumbnail, which is where the featured image will go. There is one more thing I like to add to my portfolio pieces: a URL to the website I'm showcasing. First, we'll create the function that will add this box in the admin:

```
add_action("admin_init", "jlc_projects_admin_init");
function jlc_projects_admin_init(){
    add_meta_box("jlc-projects-meta", __("Project Link"),
    "jlc_projects_options", "portfolio", "side", "low");
}
function jlc_projects_options(){
        global $post;
        if ( defined('DOING_AUTOSAVE') && DOING_
        AUTOSAVE ) return $post_id;
        $custom = get_post_custom($post->ID);
        $link = $custom["jlc_projects_link"][0];
?>
  <input name="jlc_projects_link" placeholder="http://"
  value="<?php echo $link; ?>" />
<?php
    }
```

These functions are fairly straightforward. When the admin is initiated (that is, loaded), we'll call a function called `jlc_projects_admin_init()` that will create a new meta box for portfolio items. In order to generate that box, a new function, `jlc_projects_options()`, is called.

Once inside the `options` function, we simply grab the `link` value, which we've called `jlc_projects_link`, and print it out. But first, we want to make sure an autosave isn't being performed. If it is, we will probably lose data. After that, we need to actually save the metadata when the post is saved:

```
add_action('save_post', 'jlc_projects_save');
function jlc_projects_save(){
    global $post;
    if ( defined('DOING_AUTOSAVE') && DOING_AUTOSAVE ){
        return $post_id;
    }else{
        update_post_meta($post->ID, "jlc_projects_link",
        $_POST["jlc_projects_link"]);
    }
}
```



Basic grid The HTML template used here, at full width. It's a simple header with three columns of projects

With the admin section for our Custom Post Types created, it's time to add some frontend functionality to help display our projects to visitors. This consists of an archive template, a single page template and a shortcode (not covered in this tutorial). But before we do that, there is one other function we're going to create for displaying images: picturefill.js.

This piece of JavaScript (GitHub repo at *rwdwp.com/23*) allows you to define a set of media queries to switch an image to a version friendlier to the size of the screen it is being viewed on. This also has implications for load time, as you can probably assume that a smaller screen means a mobile device using 4G, 3G, or even EDGE. I know that isn't always the case, but it's better than nothing.

You can see the markup pattern for a standard picturefill element on the GitHub repo. We can have an unlimited number of `<span>` elements for each size of the image we have. There is also a fallback for users without JavaScript. As you can imagine, since WordPress creates multiple versions of every image we upload using the Media Uploader, it lends itself nicely to picturefill.js.

The first thing we should do is load the script, which is in the '/js/' folder in our plugin's directory. We add the following code to 'jlc-projects.php':

```
function jlc_projects_scripts() {
    wp_enqueue_script( 'picturefill', JLCP_PATH.'js/
    picturefill.js', array());
}
add_action( 'wp_enqueue_scripts', 'jlc_projects_scripts' );
```

This will load our JavaScript with the scripts being loaded by other plugins. It will also ensure that we aren't loading picturefill.js more than once.

RESOURCE

Scott Jehl's original blog post about picturefill.js discusses the need for the polyfill and includes a link to a demo site: *netm.ag/picturefill-254*

As our projects will be using the featured image section to display the screenshot, we can replace the default featured image markup by using the `post_thumbnail_html` filter. Note that this function will replace all featured image sections on the site. If this causes a conflict (it probably will), you should add some conditionals to your plugin to make sure this filter is only being used on portfolio pages.

```
add_filter( 'post_thumbnail_html', 'jlc_projects_get_
featured_image');
function jlc_projects_get_featured_image($html,
$aid=false){
    $sizes= array(''thumbnail', 'medium', 'large', 'jlc_project',
    'full');
        $img= '<span data-picture data-alt="'.get_the_
        title().'">';
        $ct= 0;
        $aid= (!$aid) ? get_post_thumbnail_id() : $aid;
        foreach($sizes as $size){
            $url= wp_get_attachment_image_src($aid,
            $size);
            $width= ($ct < sizeof($sizes)-1) ? ($url[1]*0.66) :
            ($width/0.66)+25;
            $img.= '
                <span data-src="'. $url[0] .'"';
            $img.= ($ct > 0) ? ' data-media="(min-width: '.
            $width .'px)"></span>' :'></span>';
            $ct++;
        }
        $url= wp_get_attachment_image_src( $aid,
        $sizes[1]);
    $img.=  '<noscript>
        <img src="'.$url[0] .'" alt="'.get_the_title().'">
                </noscript>
            </span>';
        return $img;
}
```

Mobile view A portion of the archive template displayed on a mobile-sized screen. The cards shrink to single column with centered images

There are a few things going on here. First, the function has an array of all the image sizes in WP that we want to use. If you have your own sizes defined, you need to add them. This is so the picturefill element is accurately populated.

After some set-up (defining the image sizes, opening the picturefill element, initialising a counter), it moves through the `$sizes`, printing an image entry for each. For each entry, `wp_get_attachment_image_src()` is called to grab the URL of the image based on the image's ID (which `get_post_thumbnail_id()` returns based on the post ID) and the size. `wp_get_attachment_image_src()` returns an array that includes the image, the width, the height, and whether or not it's cropped.

There is also a bit of maths going on here, to calculate when to determine the breakpoints, as well as how to handle the thumbnail image. I'm not going to discuss this in detail here, but it's worth noting that this is an important problem to solve. Now any time we get the post's thumbnail, the HTML returned will be from our function.

If your plugin includes CSS, keep it minimal so it doesn't butt heads with the main theme

CREATING THE ARCHIVE PAGE

Next, we will create the archive template for the new Custom Post Type. This is what will be displayed by default and will serve as our main portfolio page. Note that we will not be creating the site's homepage in this tutorial, but doing so would require either a template tag or shortcode that will execute a custom Loop using `WP_Query`.

Create a new file in whatever theme directory you are using and call it 'archive-portfolio.php'. WordPress's template hierarchy is smart enough to know that, when a user is on the portfolio page, it should display the content using this template. I recommend copying 'page.php' for this template. We will simply replace the Loop portion.

I recommend that you use a template without a sidebar, or a single-column template. The CSS referenced here will work a bit more nicely. Here's what our Loop looks like:

```
<?php while (have_posts()) : the_post(); ?>
    <div class="card">
        <?php the_post_thumbnail('jlc_project'); ?>
        <h3><?php the_title(); ?></h3>
```

```
        <p><?php echo get_the_excerpt(); ?> <a href="<?php
        the_permalink(); ?>">read more...</a></p>
    </div>
<?php endwhile; ?>
```

This should be pretty straightforward. We are replacing the default HTML for `the_post_thumbnail()`, so we don't need to worry about which image to use because all sizes will be returned using picturefill.js markup. I opted to use `get_the_excerpt()` in order to exclude any markup included by `the_excerpt()`.

When designing a plugin that includes some CSS, it's important to make it as minimal as possible so it doesn't butt heads with the theme's CSS or give the user the ability to exclude your CSS completely. As we're creating templates within the theme, we have a little more wiggle room.

Here's a portion of the (Sass-generated) CSS that I've added to each project on the archive page:

```
.card img {
    display: block;
    margin: 0 auto; }
@media screen and (min-width: 45.88em) {
    .card {
        display: inline-block;
        width: 40%; } }
@media screen and (min-width: 54.62em) {
    .card {
        width: 44%; } }
@media screen and (min-width: 76.38em) {
    .card {
        width: 29%; } }
@media screen and (min-width: 99.4em) {
    .card {
        width: 30%; }
```

I've determined which breakpoints were best for placing the project cards side by side. I've also made the featured images centre automatically.

CREATING THE SINGLE PAGE

Now we'll create the single template for portfolio projects. Whenever a user visits a single project's page, this is what will display. Create a new file in your theme, call it 'single-portfolio.php' and copy another template in there (I'd recommend whatever you used for 'archive-portfolio.php'). This time we will be replacing the Loop with this code:

```
<?php while (have_posts()) : the_post(); ?>
    <h2><?php the_title(); ?></h2>
        <?php the_post_thumbnail('jlc_project'); ?>
        <?php the_content('Read the rest of this entry'); ?>
        <?php
```

Single project The single project view. The project reduces in width as the page grows to keep the text easy to read

```
            if(function_exists('jlc_projects_get_link')){
                $jlc_link= jlc_projects_get_link($post->ID);
                if($jlc_link){
        ?>
                <a href="<?php print $jlc_link; ?>"
                class="button">Visit the Site</a>
        <?php
                }
            } ?>
<?php endwhile; ?>
```

This looks similar to the archive template, but we call an extra function: `jlc_projects_get_link()`. We will add this to our plugin, and it will return the URL for the current project. In the event there is no URL, false should be returned and no button is displayed.

Here's what the function (located in `jlc-projects.php`) looks like:

```
function jlc_projects_get_link($id){
    $url= get_post_custom_values('jlc_projects_link', $pid);
    return ($url[0] != '') ? $url[0] : false;
}
```

The CSS for this page will depend largely on the theme: I've used some CSS to generate a nice button. Make sure that whatever CSS you create yourself does not interfere with the main theme.

IN CONCLUSION

By now, we've created a plugin to add a new Custom Post Type for portfolios, integrated picturefill.js to handle images better, and created two theme templates to display the information.

The GitHub repo for the tutorial contains all of the code shown here, plus the theme I used, a shortcode and a template tag.

BOOK

The issues raised in this tutorial are discussed at greater length in Joe Casabona's book, *Responsive Design with WordPress*: *rwdwp.com*

ABOUT THE AUTHOR

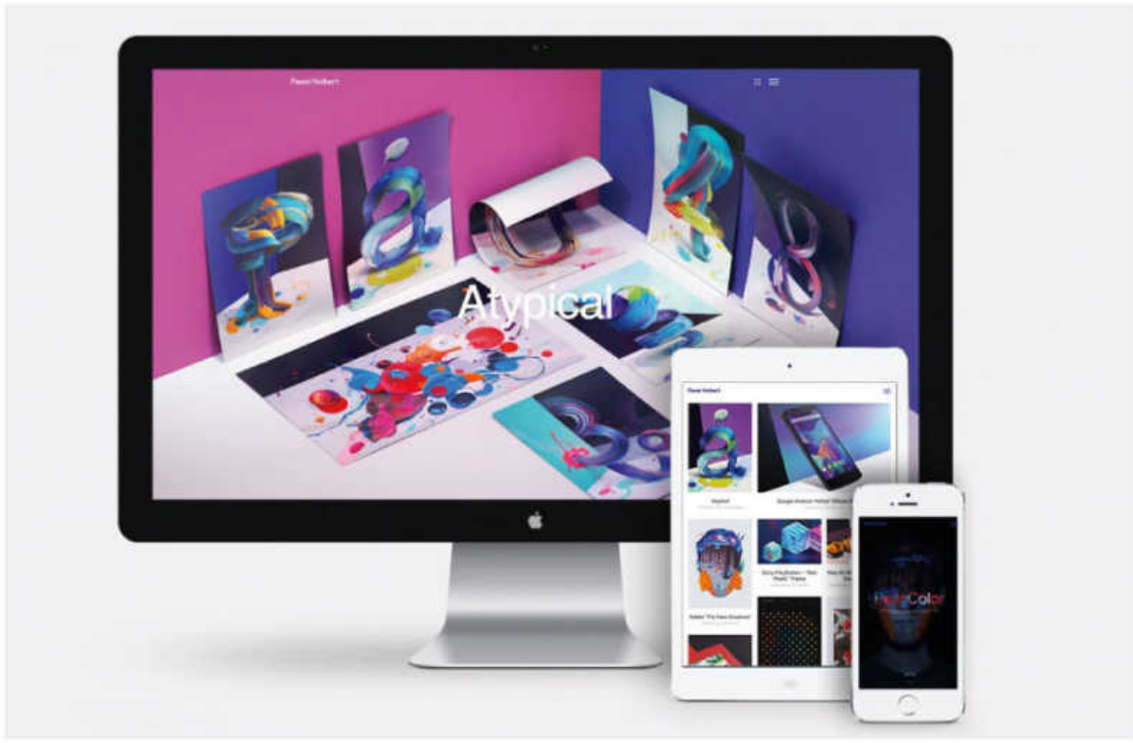
TOBIAS VAN SCHNEIDER

w: *vanschneider.com*

t: @schneidertobias

job: Creative director, Spotify; co-founder, Semplice

areas of expertise: Product design, art direction, branding



SEMPLICE

CREATE BEAUTIFUL CASE STUDIES WITH SEMPLICE

Tobias van Schneider introduces Semplice, a fully customisable case study portfolio system based on WordPress

For a very long time the way we, as creative professionals, have presented our work hasn't changed at all. We know how important a great portfolio can be for our careers and especially for how we position and present ourselves in the industry. Still, most of us struggle with the very time-consuming task of creating an outstanding online portfolio. This can be for a number of reasons. In most cases it's either a lack of technical skills that hinders us on the road to bringing our vision to life, or simply that the available tools aren't delivering the results we have in mind.

With Semplice, we aimed to create the new portfolio tool we designers have been looking for. Created by designers for designers, Semplice enables you to create beautiful branded pages that tell the stories of your project the way it should be told. What's more, you can achieve all of this without using HTML or any programming languages.

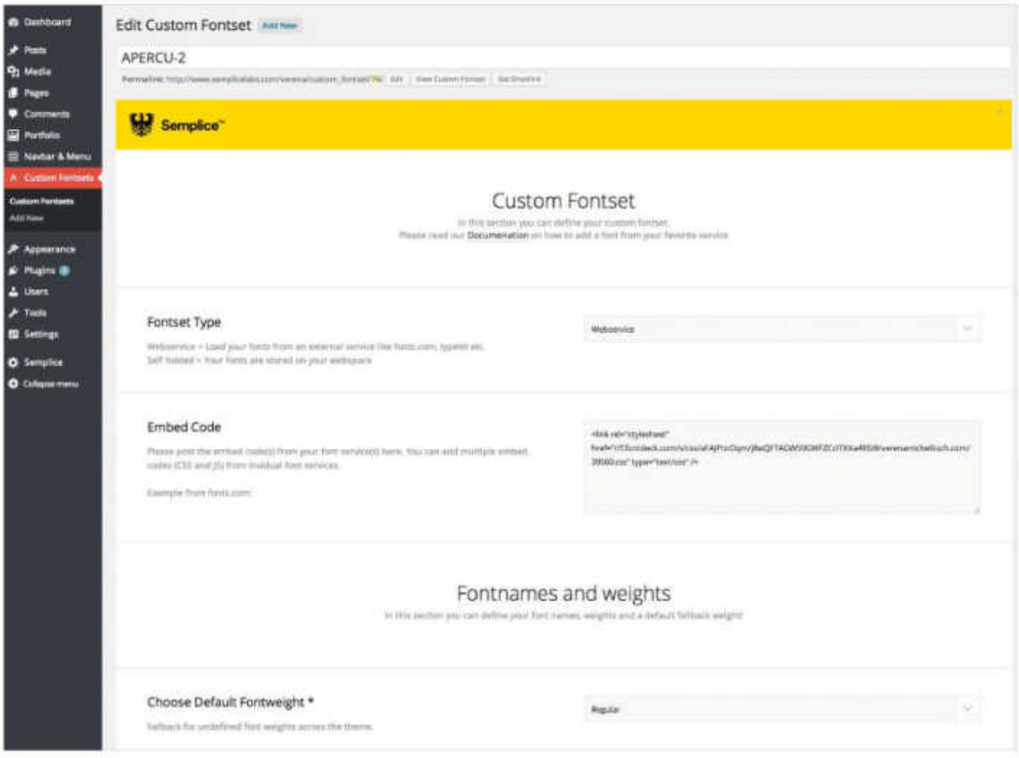
Semplice is built on top of WordPress, which means all your projects are together in one place. Everything created with the Semplice content editor is easy to edit from anywhere, which makes updating your portfolio as easy as creating it. On top of that, everything is automatically responsive, making browsing through your work on any device a joy. You focus on creating beautiful work, Semplice takes care of the rest.

01 While Semplice is built on top of WordPress (which gives you plenty of freedom if you do want to jump into code), everything has been created with a completely custom interface on top of the usual WordPress backend. For this example project, I will walk you through some of the basic functionality that will enable you to create a simple branded page that shows off a project case study.

Semplice was built as a tool for translating your ideas into reality, so we recommend having a layout concept already planned out in your mind, and then using Semplice to bring it into reality.

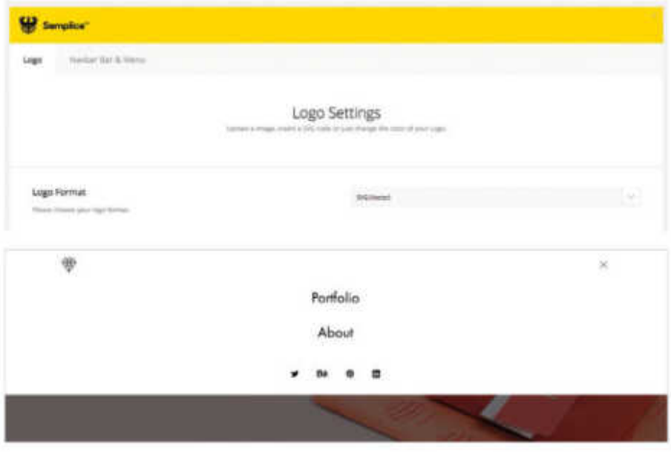
02 Each project or page is built up in a modular way, which makes the process of building your portfolio more efficient, and ensures it's easy to maintain. Let's start by adding our custom font sets. You can add as many of these as you like, from any font service you choose. For example, to achieve a fully branded effect, each project page could have its very own font set.

To add a new font set, click on `FontSets > Add New`. All you have to do is paste the JavaScript or CSS code from TypeKit, *Fonts.com* or whichever font hosting service you have selected. Once you're done, just save your custom font set. We will assign it to a specific project later.



Step 2 Add as many font sets as you like, simply by pasting the JS or CSS code from the font service you choose

03 Our next step is to add a custom navigation. Think of the navigation of a global element that you could assign to as many projects as you like. Semplice gives you the option to create fully custom navigations, which you can then assign to a specific project page or pages. Click on `Navigations > Create New` to create a new navigation and customise it in any way you want. You can add your logo, change the colours, font choice and so on. Once you're done, save your navigation – we will assign it to your project later.



Step 3 Create custom navigations and add these to your project pages

04 Now we have our basic elements in place – the navigation and font sets – we can create a new project page by clicking on `Portfolio > Add new work`, and start creating our special project page. We begin by adding basic information about the project, followed by our custom thumbnail for the homepage grid. The great thing about Semplice is that you're not limited to a thumbnail grid template as in most portfolio systems – each thumbnail item can have its own size, as long as you stick to the basic grid.

EXPERT TIP

HACKING SEMPLICE

Semplice is made for hacking. When we were building Semplice, we kept in mind that most designers would already know exactly what they wanted to create with it. Semplice gives you a blank canvas, a set of tools to play with and the freedom to do whatever you want, while Semplice ensures it all works fine on mobile devices. You own the full source code – so make it your own!

05 Now we can add some custom branding to our project page. This helps make each case study a complete, immersive experience. Click on the Branding tab at the top, where we now can select the navigation style we created earlier. As everything is modular, you could use the same navigation style for multiple different projects.

06 Our fullscreen cover gives you a great opportunity to welcome the viewer with a beautiful fullscreen video or immersive fullscreen imagery – simply click Fullscreen Cover in the top bar when creating a new project or page in WordPress. This feature is completely optional, but it might be the perfect thing to set the tone for your cast study.

07 We're now entering the core part of the experience: the Semplice content editor. By clicking on the Content Editor tab (which you'll find in the top bar after creating a new project or page), you enter a fully customised content editor on top of WordPress. You'll will see a blank canvas, giving you complete freedom to start creating your custom portfolio. You can create any layout you want, using the tools available at the top of the page. These range from images and paragraphs to galleries and multi-column modules.

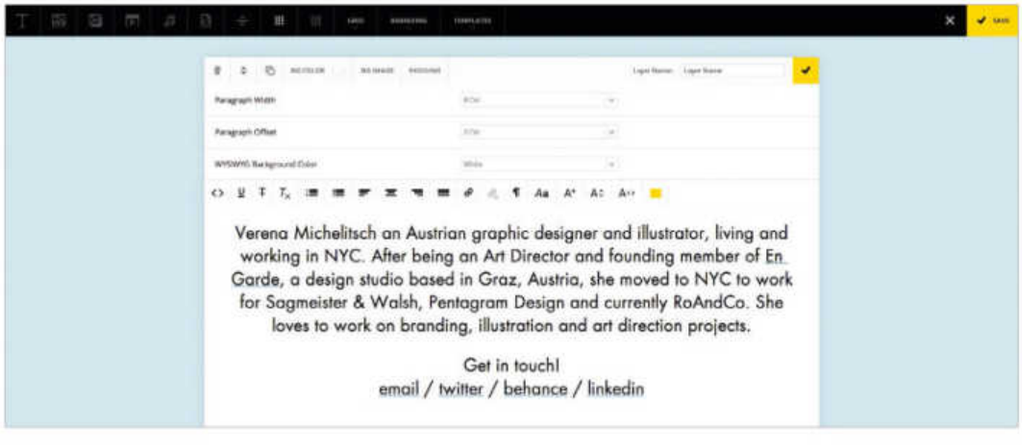
Step 6 Add a fullscreen cover to create a fully immersive experience for your users

08 Let's start by adding an introduction using the paragraph module. Just click on the 'T' icon in the top-left corner (you can mouseover the different buttons to see which modules are available). We can customise the module's background image, colour, text alignment, width and padding.

09 In the top navigation we can now select the font set we created earlier, and apply this to our paragraph layers. Once you're done, hit the checkmark icon in the top-right corner to save your layer and immediately go into preview mode.

10 At this point, you can start to see how the website will end up looking. The Semplice content editor is completely visual and can be easily accessed a outlined in step 7. Everything can be edited by simply clicking on the layer you want to change. Let's add another layer right underneath the first. We can do this by choosing the Image tool in the top left.

11 Every time you add a new piece of content, you can add a container around it which you can style in any way you like. Adding an image gives you many



Step 8 Add a project description to your case study page by including a paragraph module

EXPERT TIP

SEMPLICE VIDEOS

Semplice provides in-depth video tutorials which you can access after purchasing the software. Here, we also share our knowledge base which you can easily access at *help.semplicelabs.com*

Step 9 Use the content editor to view each page, so you can see your portfolio site take shape as you build

EXPERT TIP

SEMPLICE TEMPLATES

Save time by using templates and the Block Duplicate functionality. Each project or case study you've created can be used as a template for any new case study you want to add in the future. When you first start a new project, you can select a previous project to use as a template. This can be a great time-saver.

options – you can position and scale the image, or even define another background image to sit behind the image you just added. You can be as creative as you want to be, as Semplice keeps everything responsive and ensures your design works properly on mobile devices.

12 Now we'll add a multi-column module. With this you can essentially create any editorial layout you can imagine. You can again insert paragraphs, images, videos and more. Think of the way you lay out elements in InDesign – except now you can do it while keeping everything responsive.

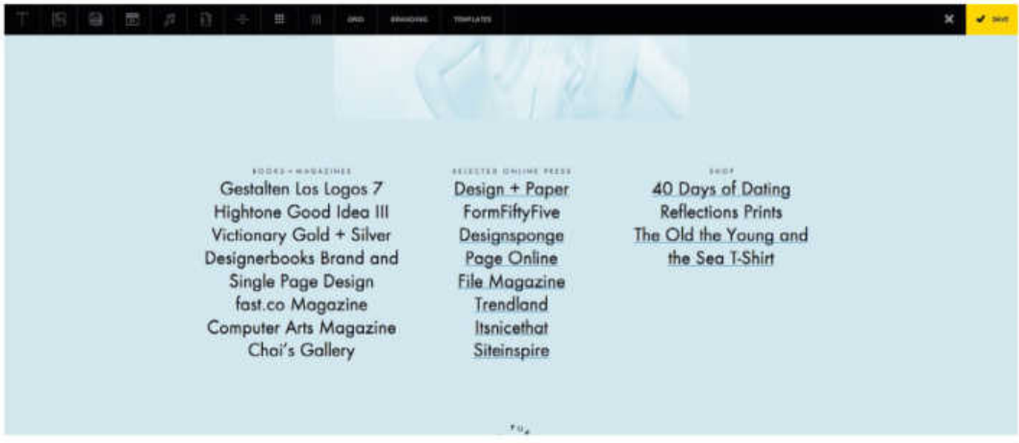
13 Every module within the content editor can be combined with any other to create custom layouts. By entering the layer mode, you can re-order your content layers by just dragging them to a different position.

14 Let's add a video from Vimeo to the page, and place a description next to it. Our default grid is made up of 12 columns. We need to split it into two sections: one side for the video and another for the short description.

15 In order to add the video to the first column, create a new column that's eight columns wide (leaving four columns for the description). The reason we do it this way is that the video will now automatically have the width of eight columns and will scale across any platform. Then we simply click on the oEmbed module for our first column and paste the Vimeo URL into the space.

16 Create a second column inside the multi-column module, and give it the width of four columns. Click on the paragraph (text) icon to add the description. By approaching the process this way, you can now easily change the video or description width by optimising the width for the column it sits in.

17 Creating with Semplice is addictive – once you've figured out all the possibilities, you can create any layout you can imagine. Semplice is more than just a tool. It's a way to make something better, to go the extra mile and tell the story of your work the way it should be told.



Step 11 Made use of the multi-column module to create any layout design you prefer

Download the files here!

All the files you need for this tutorial can be found at *netm.ag/modularGit-261*

ABOUT THE AUTHOR

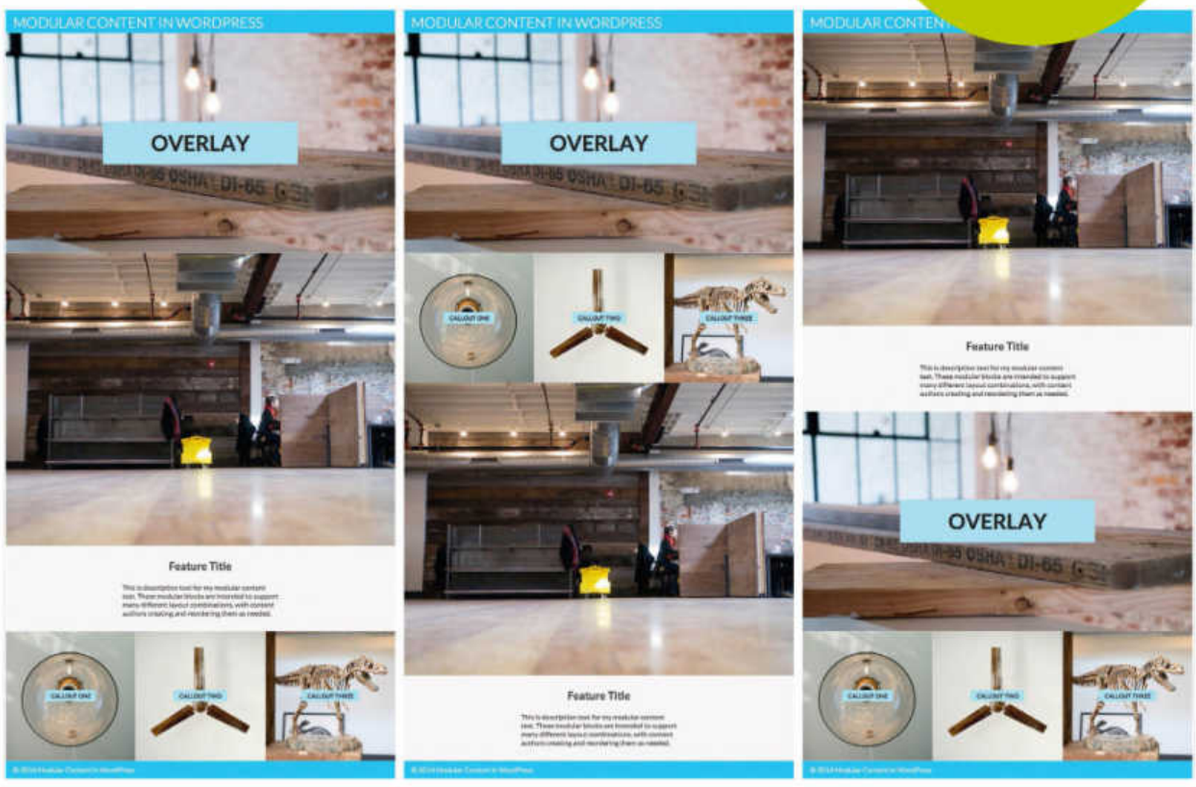
MARK LLOBRERA

w: *bluecadet.com*

t: @dirtystylus

job: Senior developer, Bluecadet

areas of expertise: HTML, JavaScript, WordPress, Drupal



ADVANCED CUSTOM FIELDS

BUILD MODULAR CONTENT SYSTEMS IN WORDPRESS

Mark Llobrera walks through how to use Advanced Custom Fields to create a WordPress CMS that is flexible as well as structurally sound

The key word that has informed my work for the last few years has been 'flexible'. A website needs to be flexible for the end user, morphing to meet them on their chosen device. But our content authors need flexibility, too, in how they create content for the site. They need to be able to mix different kinds of content to create a compelling site.

One of the ways we talk about this flexibility for both users and authors is using the term 'modular content'. Loosely defined, modular content means breaking down our content into smaller chunks that can be combined in many different ways. It is not a new idea, but it is one that is given new urgency by our new multi-screen, multi-device web. A 'page' is simply too big a unit – it must be broken down into smaller components to enable it to bend and change to better serve its authors and users.

WordPress has a few options that can be used to add some of this flexibility into a CMS. Elliot Condon's Advanced Custom Fields (*advancedcustomfields.com/pro*) has long been an essential plugin for extending the capabilities of WordPress in order to build custom CMSs. The plugin has two features that are well suited to our goals of creating a CMS that supports our flexible design

VIDEO

Mark Llobrera has created a screencast to accompany this tutorial. Watch along at *netm.ag/modularVid-261*

system, namely the Flexible Content Field and the Repeater Field.

These features allow designers and developers to create modules and craft markup to represent those modules, but they also give content authors flexibility in how they order those modules. So instead of needing many different page templates, we can use a smaller number of flexible content types and templates that are able to render a wide range of possible pages.

IDENTIFYING OUR MODULES

Let's start with an example: a product details page template. A common scenario is to have a large, evocative image at the top, supported by a section (or sections) highlighting individual features (with an optional image). Finally, callout blocks to related products could follow.

We can start by identifying our design system. There are a number of ways to do this, but at the moment I like to sketch out my site's possible templates and what they will need to support. Once I've done that, I start circling common visual

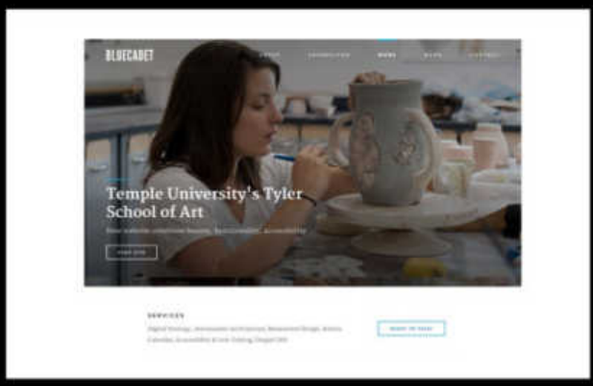
A website needs to be flexible for the user, morphing to meet them on their chosen device

structures that can support those content modules. From there I can move on to wireframes that clearly identify those modules.

In the scenario shown in the image on page 111, the template calls for three modules: A) a full-width image with text layered on top, B) a centered image with text running underneath it, and C) a series of callout blocks that have a thumbnail image with text layered on top. Our goal is a template that can support any of those blocks, in any order: ABC, ACB, BAC, BCA, ABBC and so on. Certain orders will make more sense given the desired hierarchy of the page, but we want to be able to support all options.

CREATING MODULES IN WORDPRESS

Now that we have an idea of what kinds of content we'll need to support, we can start creating our modules in WordPress. We will use Advanced Custom Fields Pro (*advancedcustomfields.com/pro*), a paid plugin that includes the Flexible Content Field and Repeater Field. Once you have the ACF Pro plugin activated, go to the new menu item `Custom Fields > Custom Fields` and add a new `Field Group`. Call it 'Core

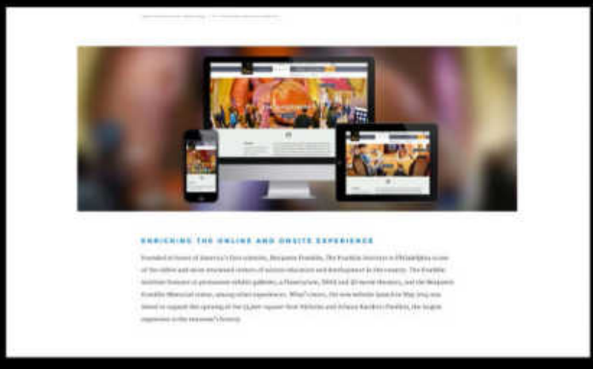


CASE STUDY

OUR TOUGHEST CLIENT: OURSELVES

At Bluecadet we've had a lot of experience building modular content systems for our clients. Our toughest client this year, however, might have been ourselves. We entered this year with the goal of redesigning our website (*bluecadet.com*), and it took a lot of hard work to create a design system that could accommodate the range of our work. For our case study pages we needed support for varying numbers and combinations of text, video and imagery. Additionally, we needed the system to be intuitive for our teammates who would be entering and maintaining content.

We ended up creating a design system with modular content blocks, including: a full-width single image, an image slideshow, a video player, a video loop, a pull quote, and regular text. Once we started building in WordPress, Advanced Custom Fields' Flexible Content Field allowed us to create each content block as a separate layout, all shared by one post type and one template file. Our content authors appreciated the ease of creating and re-ordering the different module layouts, which let them put together the right layout for each project case study page.



Mix'n'match Content authors create layouts from a range of content modules

RESOURCES

FURTHER READING

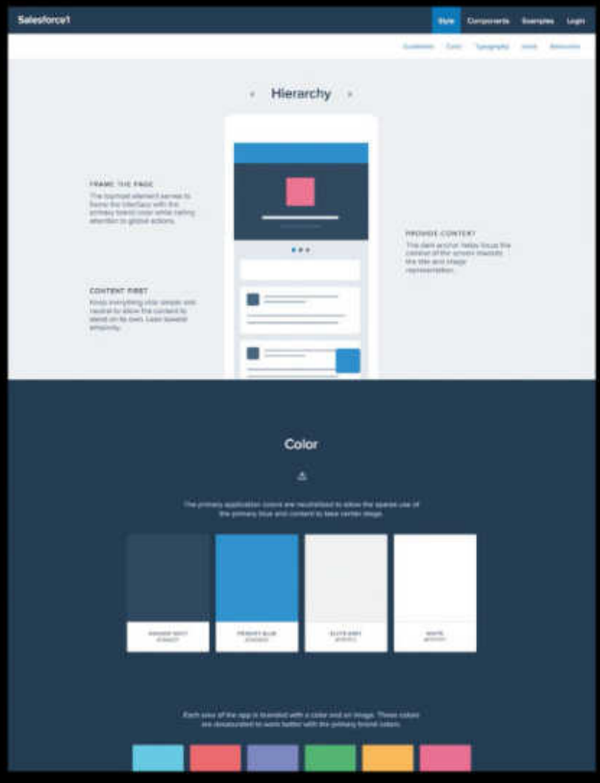
The overlap between modular content and frontend style guides is very large, and Anna Debenham has written a good overview on different approaches to building them, called 'Front-end Style Guides' (*netm.ag/style-261*). She also maintains a collection of frontend style guide resources (*netm.ag/gimme-261*).

Dave Rupert of Paravel Inc. wrote 'Responsive Deliverables' (*netm.ag/deliverables-261*) which deals with design systems, style guides and different client deliverables.

I've focused on WordPress's Advanced Custom Fields, but ExpressionEngine and CraftCMS both offer the Matrix field (*buildwithcraft.com/features/matrix*), which behaves very much like the ACF Flexible Content Field.

Karen McGrane's book *Content Strategy for Mobile* (*netm.ag/content-250*) and Sara Wacher-Boettcher's book *Content Everywhere* (*netm.ag/content-237*) both deal with identifying and structuring content, which is the starting point for any modular content system.

Newfangled's Christopher Butler wrote the post 'The way you design web content is about to change' (*netm.ag/change-261*). His article inspired me to investigate the Flexible Content Field in WordPress.



Style guide Product style guide for sales force

Modules'. Once you've created a Field Group, you can add fields to it.

Create a single field called 'Flexible Content'. Select `Flexible Content` as the field type. A Flexible Content Field gives you the option to create multiple layouts. Those modules we identified earlier? They map directly to the layouts you will create here. When you create the field it should automatically create your first layout for you.

Go ahead and label the first layout 'Image Text Overlay', with a name 'image_text_overlay'. That name is important because that's how you'll reference the field in your PHP template file. Create a field labeled 'Background Image' (name: 'background_image') and set the `Field Type` to `Image`. Add another field, labeled 'Overlay Text' (name: 'overlay_text') with a `Field Type` of `Text`. That's it for the first layout.

Once you're done with your first layout, hover over the word `Layout` in the left-hand column. Several options should options appear, including `Add New`. Click on that to create your second layout. Label it 'Image and Text Block' (name: 'image_and_text_block'). Add a field labeled 'Featured Image' (name: 'featured_image', field type: `Image`). Add another field, labeled 'Description Text' (name: 'description_text', field type: `Wysiwyg Editor`). This layout maps to the centred image with a text block underneath it.

Each layout maps to a module, so we can craft layout-specific markup in our template

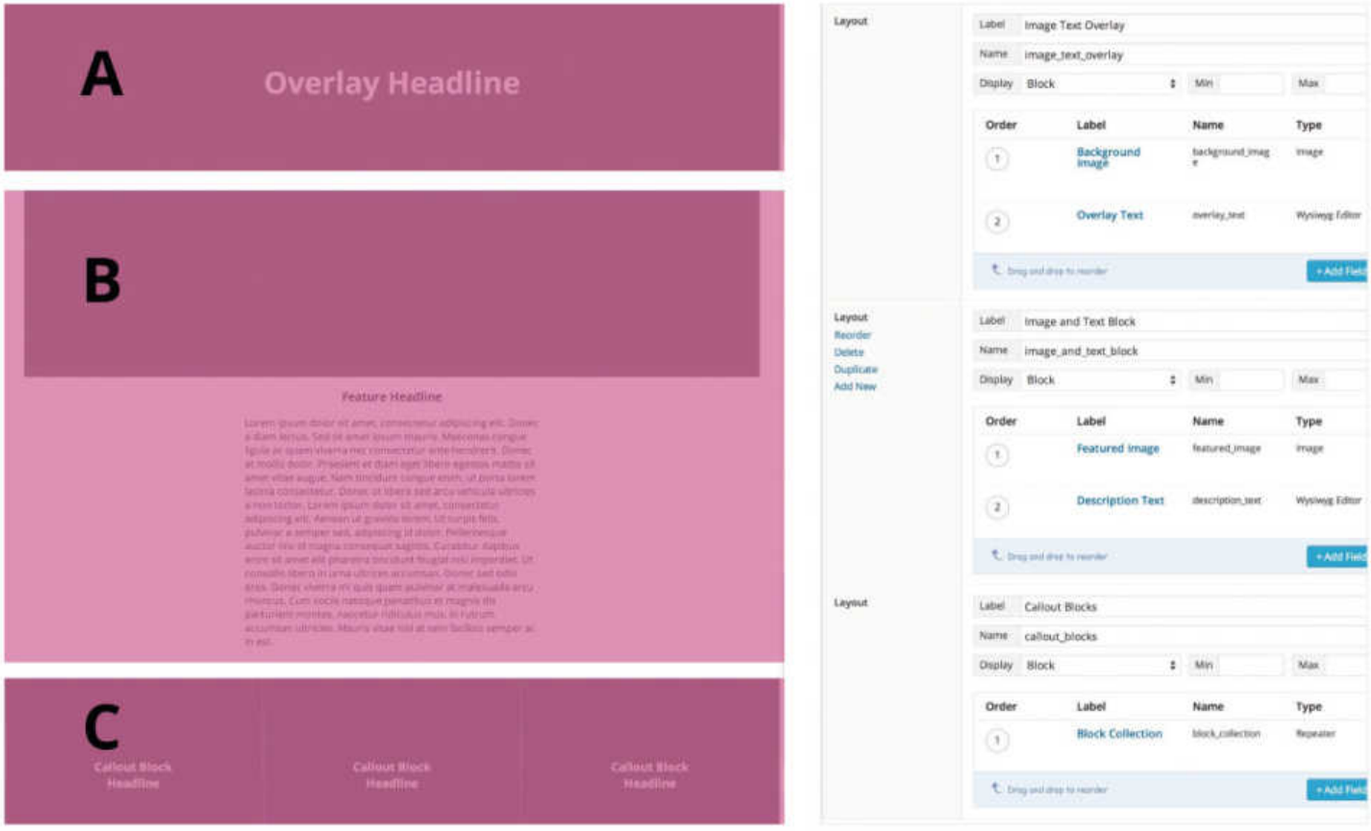
Finally, create your last layout. Label it 'Callout Blocks' (name: 'callout_blocks'). Create a new field labeled 'Block Collection' (name: 'block_collection', field type: `Repeater`). A repeater is a way to create multiple field collections.

In the `Sub Fields` section of the `Repeater` field create a field labeled 'Block Image' (name: 'block_image', field type: `Image`). Create a `Text` field labeled 'Block Text' (name: 'block_text').

FIELDS EXPLAINED

There are our three layouts. If the process of creating field groups and fields is a little confusing, the ACF website has very detailed documentation and videos showing how to create them.

The modules are all pretty similar: they contain an `Image` field, and a `Text` or `Wysiwyg` field for entering



Left A simple detail page wireframe, with three distinct visual modules identified
Right A Flexible Content Field with multiple layouts

text. The only wrinkle is that the last module ontains a `Repeater` field with sub fields, so that we can create multiple field collections within one larger layout.

You may be wondering, why create separate layouts when each layout contains very similar fields? Well, each layout maps to a module in our design system, and this allows us to craft markup in our template that is specific to each layout.

But even more important than the ability to break our markup into separate pieces in our template is the flexibility that layouts provide. Each different layout can be created any number of times. Furthermore, each layout can be dragged in the display order to give your content author the flexibility to decide the best arrangement for the content that makes up each page.

Some pages might only use one or two of the layouts, depending on what content is available for the page. By default ACF will attach your Field Group to the default Post type. If you are creating a big CMS, it's very likely that you'll attach it to a custom post type of your making.

Now let's take a look at it in action. Note the option to create any one of the three layouts by hovering over the '+' symbol. Once your layout has been created, you can also drag your fields to put them them into any desired order.

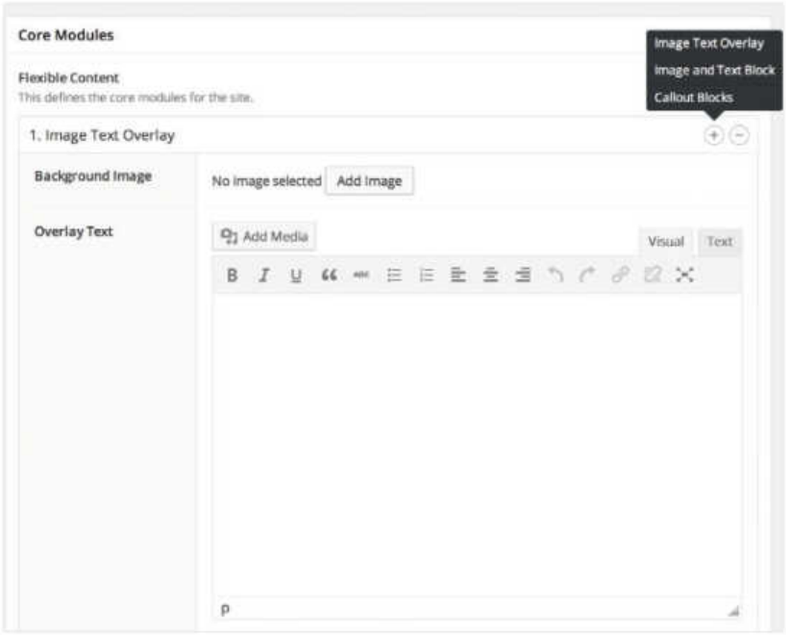
CRAFTING OUR TEMPLATE

So now that you have your content structured into these modular layouts, how do you create a template to display them? Well, it turns out that the PHP template is quite simple. In it you will check for rows in our `Core Module`'s `Flexible Content Field`.

For each row, it can then check the layout type and render the appropriate markup. For rendering the fields themselves we will use the handy ACF methods `the_sub_field()` (*netm.ag/sub-261*) and `get_sub_field()` (*netm.ag/getsub-261*).

I like to generate my themes using underscores.me (*underscores.me*). Your theme should have a `content-single.php` file already present – if not, you can create one. Delete everything in the file and replace it with this code:

```
<?php ?>
<article id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
<div class="entry-content">
<?php
if( have_rows('flexible_content') ):
```



Get adaptive Flexible Content Field layouts are easy to create and reorder

```
while ( have_rows('flexible_content') ) : the_row();
if ( get_row_layout() == 'image_text_overlay' ): ?>
<section class="text-overlay">
<h1><?php the_sub_field('overlay_text'); ?></h1>
<?php $background_image = get_sub_field('background_image'); ?>
<img src="<?php echo $background_image['url']; ?>">
</section>
<?php elseif (get_row_layout() == 'image_and_text_block' ): ?>
<section class="text-centered">
<?php $featured_image = get_sub_field('featured_image'); ?>
<img src="<?php echo $featured_image['url']; ?>">
<section class="text-block">
<?php the_sub_field('description_text'); ?>
</section>
</section>
<?php elseif (get_row_layout() == 'callout_blocks' ): ?>
<?php if( have_rows('block_collection') ): ?>
<section class="text-callout">
<?php while( have_rows('block_collection') ): the_row(); ?>
<?php $block_image = get_sub_field('block_image'); ?>
<section class="text-callout-block"><h1><?php the_sub_field('block_text'); ?></h1><img src="<?php echo $block_image['url']; ?>"></section>
<?php endwhile; ?> <!-- block collection while -->
</section>
<?php endif; ?> <!-- block collection if -->
<?php endif; ?>
<?php endwhile; ?>
<?php endif; ?>
</div><!-- .entry-content -->
</article><!-- #post-## -->
```

This template begins by checking if the field `flexible_content` has any rows. If it does, then it loops through each row of the field. Each time through, it checks the `Row Layout` using the `get_row_layout()` method.

You can see that the three names it is checking are the ones you specified for the three layouts: `image_ text_overlay`, `image_and_text_block`, and `callout blocks`. Each of those layouts gets its own (modular) piece of markup. In each markup block you then call `the_sub_ field()` (or `get_sub_field()`, in the case of images that you'll need to set to a variable for printing later).

Now you have a template to render the correct markup for each of your modules, you can style the results with CSS. In many cases I prefer to work out the CSS for each module in a frontend style guide or static HTML mockup, so this step normally involves copying, pasting and testing into the WordPress theme CSS file.

Content authors are given intuitive tools to create layouts and drag them into any order

The CSS I have used is included in the tutorial files. It is fluid (to an extent) but not responsive – you are encouraged to play with it and come up with your own markup/CSS patterns once you understand how the ACF layouts and fields are rendered.

THE FINAL (FLEXIBLE) RESULT

The end result is a page template that supports a flexible design system, on both the authoring side and the rendering side. Content authors are given intuitive tools to create layouts and drag and drop them into the desired order. The result will be well-crafted, consistent markup, ultimately giving the site's users a better experience.

11 REASONS WHY YOU SHOULD ATTEND

generate LONDON

Grand Connaught Rooms : 17–18 September 2015

The conference for web designers, presented by net and Creative Bloq, is back! Here's why you need to be there ...

01. An outstanding speaker line-up
We've put together a stellar bunch of people for this event (see *netm.ag/speakersLondon*)

02. Two days, one track
We've expanded the conference to two days, one track, so you won't miss anything!

03. Tomorrow's trends today
Learn the principles and techniques you need to stay ahead of the curve

04. Shopify's interactive workshop
Get an overview of the Shopify platform in this 30-minute session

05. Superb networking opportunities
Make contact with fellow web folk in a fun, energised environment

06. An amazing location
Generate will take place at the Grand Connaught Rooms, just off Covent Garden

07. Hang out with the speakers
Why not share a beer with the speakers and the Generate team the day before the show?

08. Party the night away
Don't miss our party at the end of day one, for more networking opportunities

09. The net awards
The net awards ceremony is taking place on the evening of day two. Celebrate with us!

10. Exclusive Generate videos
Afterwards, you'll get exclusive online access to all the talks you may have missed

11. Did we mention lunch?
If you still need persuading, here's a final clincher – the lunch will be fantastic!

ERIC MEYER
meyerweb.com

RACHEL ANDREW
rachelandrew.co.uk

SARA SOUEIDAN
sarasoueidan.com

JOHN ALLSOPP
johnfallsopp.com

TICKETS ON SALE NOW

Explore CSS, UX, web performance strategies, the Internet of Things and much, much more!

generateconf.com/london-2015

View source files here!

All the files you need for this tutorial can be found at *netm.ag/duckgit-271*

ABOUT THE AUTHOR

KIRSTY BURGOINE

w: *kirstyburgoine.co.uk*

t: @kirstyburgoine

job: Web developer, Employee of the Month (every month) at Kirsty Burgoine Ltd

areas of expertise: WordPress development, frontend development

ADVANCED CUSTOM FIELDS

CREATE A DUCK RACING SITE WITH ACF

Kirsty Burgoine walks through how to use Advanced Custom Fields to create a website for rubber duck racing

The introduction of both Advanced Custom Fields and custom post types to WordPress gave us, as developers, a massive amount of flexibility we never had before. It was the dawn of a new era (cue dramatic music) and moved WordPress away from being just another blogging platform into a fully fledged CMS system.

These days there are a variety of ways you can create custom fields. My personal choice, and what I'm going to use in this article, is the Advanced Custom Fields plugin by Elliot Condon: *advancedcustomfields.com*.

So what are we going to use this for? Imagine there was a league for rubber duck races (not all rubber ducks are for debugging, you know!). We would obviously need a website that showed all of the upcoming races and the important details about each race, such as the date and time. And then to make things a bit more fun we would also like to know which ducks will be in each race, and if you fancied a flutter, a way to figure out how much money you could win if you bet on certain ducks.

That is what we're going to make in this tutorial. You can take a look at the finished product at *netm.ag/duckdemo-271*.

WHAT YOU WILL NEED

I'm going to assume you already know how to install WordPress and create content. I'm also going to assume you have a working knowledge of developing WordPress themes, specifically using parent and child themes, and the basics of creating custom post types and your own plugins. This means we can focus on creating and using custom fields. You will need a clean install of WordPress to work with, the Twenty Fifteen theme installed (*wordpress.org/themes/twentyfifteen*) and a child theme for it activated.

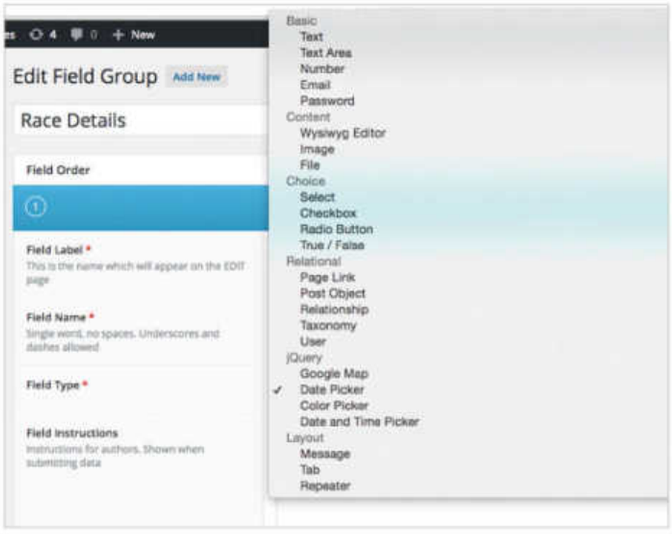
You will also need to install the ACF plugin and the repeater field add-on.

SETTING UP THE CUSTOM FIELDS

Once we have custom post types for both ducks and races, we will need to set up some custom fields for the races. Creating the custom fields is very easy using ACF.

1. Go to Advanced Custom Fields at the bottom of the main menu in the WP Admin, and create a new group
2. Give the group a name. I've called this group 'Race Details'
3. In the fields box, create some fields to store the additional data. In this case that is: date (date field), time (standard text field), ducks (repeater field)
4. In the repeater field we would create the following sub-fields: duck (post object field) and odds (number field)
5. The final step is to set the group to only display in the admin if the post type is 'races'

Once we've done that we can enter some race details, and then create some ducks and add them to that race. Then we can move on to the fun stuff.



Left The many field type options available out of the box are one of the reasons why the ACF plugin is so powerful

Right As well as using the native functions for the post, with ACF we can display additional details, like the date and time of the race

THE CODE BIT

Working in our child theme, first we need to add the date and time details to the race page. Create a new file in your child theme called 'single-races.php' (providing you called your races post type 'races'). To retrieve the date and time information, add the following to the top of the page, just inside the loop:

```
$race_date = get_field('race_date');
$race_time = get_field('race_time');
$date = DateTime::createFromFormat('Ymd', get_field('race_date'));
```

And then display the details:

```
<p class="meta">
<strong><?php echo $date->format('l jS F Y'); ?></strong>
<br />
<?php echo $race_time; ?>
</p>
<hr />
```

That was easy, right? Using the `get_field();` function, we can get the value of our custom field and then simply echo the value.

Using the date field gives us plenty of flexibility. For example, we could amend the main `WP_Query` so that we publish the posts in event date order instead of publish date. Alternatively, we could set them to only display if the date is current or in the future, or we could create a nice jQuery calendar displaying events on any given day at a glance. For an example of this, take a look at the code for the upcoming race on the homepage of the demo site: *netm.ag/demo-271*.

DISPLAYING THE DUCKS

This is where ACF really starts to come into its own. In this instance what we want to do is list all the ducks that will be in this particular race in a table, along with the details stored for each one (the duck's profile picture, the name and the odds for each).

We do this by using `if ( have_rows('the_ducks') ) : while ( have_rows('the_ducks') ): the_row();` (`'the_ducks'` being the name of our repeater field) to loop through the ducks in much the same way as the standard WordPress loop we all know and love.

First we see if there are any rows within the `the_ducks` repeater field:

```
if ( have_rows('the_ducks') ) :
```

... which you will notice is almost the same as:

```
if ( have_posts() ) :
```

If there are any rows in our `the_ducks` repeater field, we set up our table, and using `while` we loop through the rows, in a similar way to how we would loop through posts with our standard query. Once we're inside the loop we need to get our data and display it in each row of our table:

```
$ducks = get_sub_field('duck');
  $odds = get_sub_field('odds');
```

Because we are now inside a repeater field, instead of using `get_field()` we need to use `get_sub_field()` . Displaying the odds is simply a case of echoing `$odds` because it is a numbers field.

However, `$ducks` stores the post object for each duck custom post selected. This means we need to ▶

IN-DEPTH

CUSTOM POST TYPES AND PLUGINS

So that we can show information about each race in the season, as well as information about the individual ducks, we will need to have two clear types of information available on our site. Therefore we will need to create two custom post types: one for the races and one for ducks.

In order to keep the different content types separate, I like to put the code for creating each custom post type into its own plugin. This has a number of benefits. Probably the most important benefit is that it ensures none of our content is reliant on the theme. One of the great things about WordPress is how easy it is to change themes. By keeping all of our code for CPTs in plugins instead of our theme's functions file, if the theme is ever changed, the additional post types are not lost and can easily be transferred to the new theme.

Another benefit is that, even though the website will require the ACF plugin and any add-ons to be activated, exporting the code for each group of custom fields to PHP, and including that code in the custom post type plugin as well means all the code relating to that content type is kept together. Plus, it's always handy to be able to switch post types on and off.

On their own Setting up custom post types so they are their own plugin means you'll still be able ot use the content if the theme is changed

use `setup_postdata()`, so we can loop through each duck and display its profile picture, name and odds using the standard WordPress hooks.

```
<?php if ( $ducks ) :
$post = $ducks;
setup_postdata( $post ); ?>
<tr>
<td class="pic">
        <?php if ( has_post_thumbnail() ) :
                the_post_thumbnail('duck-thumb');
        endif; ?></td>
        <td class="name"><?php the_title(); ?></td>

        <td data-duckodds="<?php echo $odds; ?>"
id="theodds" class="theodds"><?php echo $odds; ?> / 1</td>
</tr>
<?php    wp_reset_postdata(); // IMPORTANT - reset the
$post object so the rest of the page works correctly
endif; ?>
```

We now have the race details displayed, and the ducks in the race listed. Let's have some fun …

PLACE YOUR BETS

If we add a couple of fields to the HTML table, with a little bit of jQuery black magic and using the `$odds` custom field we set up, we can allow people to work out what they could win if they placed a bet on any of the ducks.

If we add the following table cells to each row, we get two new columns – an input field with a default value of £10, and the calculated winnings:

```
<td class="money">&pound;<input type="text"
name="betting-amount" value="10"></td>
<td class="winnings">&pound; <span class="js-winnings"></
span></td>
```

Now place the jQuery magic in the 'footer.php' file in your child theme, before the closing `<body>` tag:

```
<script type="text/javascript">
function calculate_odds(row) {

        var winnings = 0;
        var odds = $(row).find('.theodds').attr('data-
duckodds'),
        betting_amount = $(row).find('[name="betting-
amount"]').val()
        winnings = (parseFloat(odds) / 1 + 1) * betting_
amount;
$(row).find('.js-winnings').html(winnings.toFixed(2));
}

$(document).ready(function() {
$('.lineup tbody tr').each(function() {
                calculate_odds();
                $(this).find('[name="betting-amount"]').
on('keyup', function() {
                        calculate_odds($(this).
closest('tr'));
                }).trigger('keyup');
        });
});
</script>
```

Now the grand unveil … ta-dah! We have created a nice page displaying information about the race. We can also change the betting value for each duck, and the potential winnings will update automatically!

ABOUT THE AUTHOR

SAM HERNANDEZ

w: *samhernandez.me*

t: @sam_h

job: Developer, Vector Media Group

areas of expertise: Full-stack development and CMS implementation

HEAD TO HEAD

WORDPRESS VS CRAFT

Sam Hernandez pits CMS titan WordPress against rising star Craft, and considers best use cases for each of them

WORDPRESS	CRAFT
Since its launch in 2003, WordPress (*wordpress.com*) has grown into the web's most popular open source blogging platform, with over 60 million users. With the help of third-party plugins it becomes a powerful CMS capable of driving everything from personal sites to BBC America. It's one of the most requested CMS solutions.	Craft (*buildwithcraft.com*) is a rising star in the commercial CMS space. Created by Pixel & Tonic, it's packed with powerful content management tools and boasts an elegant end-user experience. It comes in three flavours, depending on your needs: there's a free option for personal use, and paid-for Client and Pro versions.
UNBOXING	
WordPress ships with several themes pre-installed, as well as a third-party theme and plugin installer in the dashboard. You can have a site up and running in no time!	Craft is tuned especially towards custom design builds. It comes with an impressive array of field types and other features that, in many cases, eliminate the need for third party add-ons.
USER EXPERIENCE	
WordPress has one of the most user-friendly dashboards out there. Odds are pretty good that you or someone you know already knows how to use it, and likes it a lot.	Craft's dashboard is elegantly spartan and remarkably familiar from the first click. Its killer editing feature is the 'Live Preview' editing mode, which displays content changes in real time.
DEVELOPER EXPERIENCE	
WordPress makes it easy to extend existing themes for quick results. There's a definite learning curve to custom theme development, but there's plenty of help out there too.	Craft's blank slate approach allows developers to meet complex design and business requirements through powerful content structuring tools and a stellar template system.
SUPPORT	
WordPress has an enthusiastic, loyal community with plentiful discussion forums and help sites. The documentation is great but, if you're stuck, friendly help is never far away.	Craft includes vendor support right from the administration panel. You'll also find an active Stack Exchange site as well as a vibrant, welcoming community in Craft's Slack channel.

VERDICT

If you want to establish a web presence quickly with a proven platform, WordPress is for you. If your business is in custom-designed websites, then Craft is built intentionally with you and your clients in mind. If you haven't yet, give Craft a try. It's a great compliment, or alternative, to WordPress in your CMS toolchest.

FACT FILE

CREATORS
WordPress was founded by Matt Mullenweg and Mike Little, and is now managed by the WordPress Foundation. Craft was created by Brandon Kelley and Brad Bell of Pixel & Tonic.

ALTERNATIVE
Drupal is a popular CMS often preferred by higher education and government organisations: *drupal.org*

ABOUT THE AUTHOR

SHANNON SMITH

w: *chroni.ca*

t: @cafenoirdesign

job: Founder, Café Noir Design

areas of expertise: Multilingual web development, accessibility

ACCESSIBILITY

BUILD A MULTILINGUAL SITE WITH WORDPRESS

Take your site to the world. **Shannon Smith** walks through the components you need for a successful multilingual site

People often assume that English is the de facto language of the web, but the majority of internet use now occurs in a language other than English (*netm.ag/stat-271*). As more and more people join the online world from Russia, China and the developing world, that lead will grow.

For businesses adapting to a global market, as well as organisations attempting to foster customer loyalty at home, operating a website in multiple languages at the same time is becoming more common. With that in mind, let's take a look at techniques you can employ in WordPress to create a multilingual site.

01 MULTILINGUAL VS LOCALISED

Many people confuse a multilingual website with a unilingual website that functions in a language other than English. This is called a localised website.

Building a localised site with WordPress is relatively easy. You can currently use WordPress in any one of 55 different languages. And if you don't see your language, it's probably one of the 148 languages currently being translated. WordPress also works with a number of alphabets, including Latin, Cyrillic and Arabic, and can handle languages that are written from right to left.

02 MULTILINGUAL VS MULTIREGIONAL

Geo-localisation is another term that tends to get confused with multilingual. Often large corporations build separate websites for each country in which they operate and operate those in different languages. It's multi-regional, but that isn't quite the same as multilingual. Language use, geography and politics do not overlap perfectly.

03 STRATEGIES FOR CONTENT

There are a few ways of organising your website for multilingual content. The quick and dirty way is to add an automated translation plugin (Google AJAX translation is one) or an automated translation tool like Google Translate. This is easy and free, but the translation is not reliable and not very professional. However, it's definitely a practical option for a small personal blog.

Other websites opt for a partial solution. Some bilingual bloggers build a site in a single language but then write posts in alternating languages, occasionally translating content themselves.

However, a professional site will need a more robust strategy. The interface will need to function in multiple languages and the content will probably be largely mirrored. Human translation is a must.

Left The language pack interface

Right It's very easy to add Google Translate to a website

04 TECHNICAL CONSIDERATIONS

Before getting started, there are a number of questions you'll need to consider. In which languages will you serve content? What about the interface and the admin panel? You'll probably need a way of switching between languages. You might also be considering some form of language persistence, often via a cookie so visitors don't have to choose a language on each visit. You'll need to think about your URL structure. Plus, you'll have to consider the alphabets you'll be using, and text direction.

In areas without a dominant language, you can't have a single default language

05 POLITICS

You'll also want to think about the politics of language use. For example, many language switchers use flags, but this is not really recommended.

Language use and nation-states are not perfectly overlapping entities and it's easy to offend people when you make that assumption. You might also want to look at cultural considerations in terms of design elements like colour choice.

A third consideration is to choose a default language. Sometimes this is easy, but in areas where there isn't a dominant language (Switzerland), or where language choice is contentious (Canada, Belgium), you can't always have a single default language. A common option in Canada, for example, is to have a splash page with a language choice (though, in all honesty, it would be more accurate to describe it as a multilingual homepage). It's certainly not ideal for SEO purposes, but it's undeniably a lot better than ending up with a bunch of angry visitors.

06 WHAT ABOUT SEO?

Google has some detailed information on what it expects to see with multilingual sites. It wants the language to be obvious. That means not mixing languages on a single page. Google also suggests blocking automated translations from search engines, since they can be so unreliable that they confuse search engine crawlers.

Google suggests separating languages by URL, either by using different subdomains (*de.example.com*, *en.example.com*), subdirectories (*example.com/en*, *example.com/es*) or entirely different domains (*example.com*, *example.fr*).

Google also wants the language to be easily discoverable. Each page should be cross-linked to the other language versions. Using cookies to save language choice isn't recommended, and using JavaScript to do the same can be tricky, unless the JavaScript degrades gracefully.

Google discourages automatic redirection based on the visitor's browser language or by geo-mapping IP addresses, since this also blocks search engines. That makes language persistence difficult to implement as well. Google also ignores language labels in your site's HTML.

07 WHAT ABOUT THEMES?

You will need a WordPress theme that has been localised or internationalised (many commercial themes have been), or you'll need to localise your own theme. The WordPress Codex contains information on how to localise WordPress itself,

but the information included here is also applicable to themes.

The first step is to make your theme translatable. WordPress uses the GNU GetText localisation framework to translate the WordPress core. It can also be used in themes.

Every snippet of text in your theme needs to be wrapped with `GetText` calls. This means the snippet is passed by one of two PHP functions. `__()` is used when the snippet (called a message in the codex) is passed as an argument to another function. `_e()` is used the rest of the time. You also need to include a `text-domain`, which functions to identify the snippet as belonging to your specific theme. That means that your theme will include bits of text that look like this: `__('message','theme-name')`.

Next, you'll need to create a PO (Portable Object) file for each language you plan to use. Commercial themes often already come with .po files for popular languages. If not (or if you are using a custom theme), you can use a tool like poedit to generate a file with all the terms wrapped in `GetText` calls. Some blank themes like Roots include several language PO files and you can use these to get started.

After you have PO files, you'll want to create MO (Machine Object) files that speed up the GetText translation process. Tools like Poedit (*poedit.net*) can do this for you as well. More details can be found at *netm.ag/tools-271*.

08 THE MULTI-INSTALLATION METHOD

Once you have some translated content and a localised theme, you'll need to start building your site. One way to do this is by creating a network of sites, one in each language. If you want, you can use the multisite functionality built into WordPress. The WordPress codex has information on activating this feature, but it's quite simple. The only plugin you need to build a multilingual site with this option is a language switcher. If you are using only two languages, you can use the Bilingual Linker plugin (*netm.ag/linker-271*). Otherwise you can use the Language Switcher plugin (*netm.ag/switcher-271*), though this stopped being maintained back in 2010.

If you decide to go with the multisite option, you can use the Multisite Language Switcher plugin (*netm.ag/language-271*). There are several advantages to this solution: it is very stable, and it allows for the content to be identical across languages (or not).

However, it can mean more maintenance. You can reduce this by using the multisite feature so you only have to update one WordPress installation and one theme. You can also use child themes if you need to modify colour schemes – for cultural reasons, for example. But you will still need to manage content across several sites.

The WPML plugin is easy to use and makes it simpler to maintain the content of your site

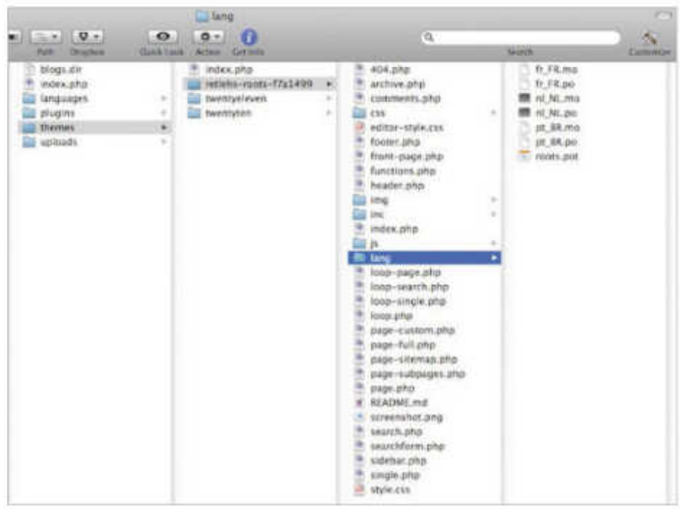
It's also worth noting that some shared hosting providers do not permit multisite installations on their servers. If you aren't using multisite, you could try the ManageWP Worker plugin, which enables you to access all your sites through one dashboard (I haven't tried this, though).

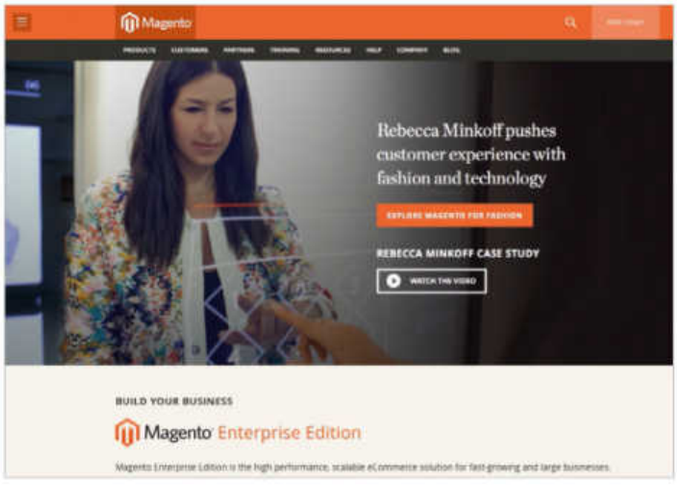
09 THE SINGLE-INSTALLATION METHOD

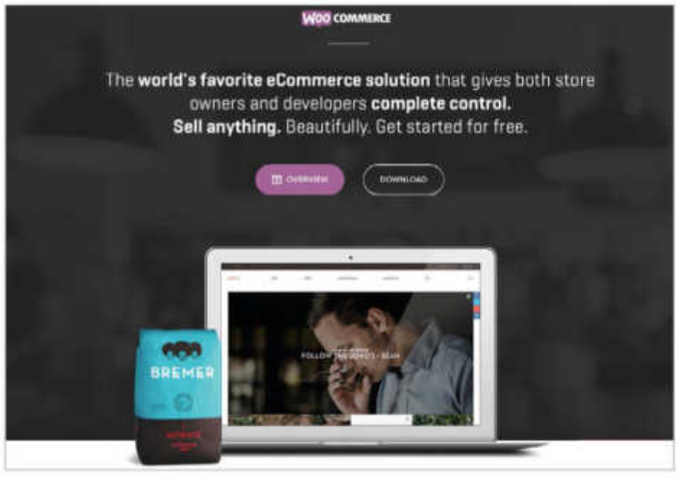
Another solid option is to build a single WordPress

Left Multilingual homepages are common in coutries like Canada

Right The Roots theme contains POT, PO and MO files for multiple languages







Left Magento is an ecommerce solution that offers multilingual functionality

Right WooCommerce is another option for ecommerce if you are using WPML

site and use a multilingual WordPress plugin. There are several plugins available but the WordPress Multilingual Plugin (*wpml.org*) is arguably the best. There are a number of options, but if you want to translate your menus (probably!) and use the latest version, you'll need to pay for the $79 CMS commercial version.

The plugin enables you to translate the WordPress posts and pages that make up your site. It creates separate posts and pages for each language and lets you link them to one another. It also enables you to translate the menus and widgets, as well as plugins.

There are other plugins that work in a similar way, but they have one major drawback: they write all the languages to the same database tables, and then show only one version of the content to the visitor at a time. That means that if you ever decide to uninstall the plugin, you'll need to manually remove all the other language versions from each WordPress page and post.

WPML With a single installation, you can operate a site in multiple languages

The single-installation method solution has advantages. The WPML plugin is very easy to use and to install. It makes it easier to maintain the content of your site, as you only need to use one admin panel and can easily jump from one language version to another. You can translate plugins that aren't available in your language. In fact, almost every part of your site can be translated using the plugin interface. However, some more complex plugins – especially ecommerce plugins – won't work with WPML, and it can be buggy. There are a few built-in troubleshooting tools, but they are hard to find and are not especially user-friendly.

10 ADVANCED FEATURES

Just because your site is multilingual doesn't mean it needs to be simple. I have not personally tried all of the following tools and plugins in multilingual installations, but I'm listing them here to give you an idea of what is possible.

Ecommerce can be tricky with a multilingual site if you need to keep track of inventory, for example. You can integrate WordPress with Magento or PrestaShop, which are multilingual. If you are using WPML, you can use WooCommerce.

If you need a fully bilingual email marketing solution, you can do that too. CakeMail offers a fully bilingual product, though it is a little pricey. If you use Campaign Monitor, you can use multiple sub-accounts for each language. If you need complex forms, both the online service Wufoo and the WordPress plugin Gravity Forms can provide those. There are also a number of solutions for multilingual forums and wikis, though often you will need multiple installations of each.

Hopefully, this gives you an idea of how to get started creating a multilingual site in Wordpress!

ABOUT THE AUTHOR

JAMES STEINBACH

w: *jamessteinbach.com/blog*

t: @jdsteinbach

job: Frontend architect

areas of expertise: Sass, CSS animation, RWD, WordPress

SASS

SWITCH FROM CSS TO SASS IN WORDPRESS

James Steinbach shows you how to combine WordPress and Sass to organise your code and streamline your workflow

Every good developer knows to look for ways to reduce repetitious work and let their computer handle mundane, mindless tasks. CSS preprocessors like Sass give us several valuable tools to help automate the frontend coding process. For example, with Sass we can use variables. So instead of running a 'find and replace' command through a long CSS file to tweak a colour value, we can simply change the variable definition.

Sass also allows us to write functions to generate blocks of repeated style code. For example, a `button` function could accept the colour or style as a parameter and generate all the standard CSS for a site's button UI: border radius, gradients, text colours and so on. We can also break up our giant stylesheets into organised modules. Nearly every CMS, plugin and web app uses directories and partials to improve code maintainability: Sass allows us to do this with our CSS.

The techniques explained in this tutorial are specific to Sass using the SCSS syntax (*netm.ag/syntax-265*), but they are applicable to most other preprocessors – like Less (*lesscss.org*) or Stylus (*learnboost.github.io/stylus*) – as well.

CONVERTING THEME STYLESHEETS

If the WordPress theme you're using has Sass files included already, the process of converting existing

VIDEO

James Steinbach has created an exclusive screencast to go with this tutorial. Watch along at *netm.ag/wordpress-265*. You'll find the gist at *netm.ag/gist-265*

Theme time The Underscores WordPress theme offers an option for users to download a version of the theme that includes Sass partials already

theme stylesheets to Sass is done for you. I typically start new themes with the Underscores template from Automattic (*underscores.me*). When creating a new theme package from its website, you'll find a '_sassify!' option if you click the 'Advanced Options' link. This will provide Automattic's default Sass library for you when you download the blank theme.

If the theme you're building or modifying doesn't include Sass files, however, you'll begin by converting the existing CSS file to Sass. This step reveals a major advantage to using Sass (SCSS): the Sass compiler reads plain CSS with no trouble at all. All you need to do to use the original `style.css` as Sass is to duplicate it and rename it `style.scss`.

SETTING UP SASS PARTIALS

Earlier I mentioned that Sass improves our workflow by letting us break thousands of lines of CSS into modular files called partials. Let's work through this step before getting to compiler methods.

If you've converted an existing CSS file to Sass, all you've got so far is a `.scss` file that's just as unwieldy as the CSS file it came from. At this point, you can start using variables and writing mixins or functions, but the Sass isn't modular or organised yet. You want to group your CSS file into sections that serve individual purposes. The goal is to be able to work out where code is found based on its file name.

Some modules of code that you'll probably want to isolate include navigation, typographic styles, sidebar widgets, footer and a grid system (although that list is not comprehensive by any means, it's just a starting point).

You will then cut and paste each modular section of CSS into its own Sass partial. A Sass partial's file name always starts with an underscore (`_`).

FOCUS ON

COMPILING SASS WITH COMPASS

Compass (*compass-style.org*) makes Sass configuration simple by storing output configuration in the `config.rb` file:

```
# Set this to the root of your project when deployed:
http_path = "/"
css_dir = ".."
sass_dir = ""
images_dir = "../images"
javascripts_dir = "../js"
# You can select your preferred output style here (can be
overridden via the command line):
# output_style = :expanded or :nested or :compact or :compressed
output_style = :expanded
# To disable debugging comments that display the original location
of your selectors. Uncomment:
# line_comments = false
```

I keep `config.rb` in my `scss` folder. In this file, the first five options are file path and directory values. The first option (`http_path`) is the project's path on the server. Next, `css_dir` specifies the directory where compiled CSS files should be saved. In this case, it's up one level from 'config.rb' in '\scss\'. The `sass_dir` option points to the directory where Sass files are found (same as `config.rb` here).

The `images_dir` and `javascripts_dir` values specify where images and JS files are kept in this project. The `output_style` option controls how the compiler generates CSS. I toggle between `:expanded` for development and `:compressed` for production. Once we save all the project options in 'config.rb', all we need to do is navigate to the folder that contains that file and run `compass watch` in Terminal or Command Line. Compass uses all the options in the config file and runs Sass to compile our '.scss' files properly.

Mastering Compass Using Compass with a config.rb makes it easy to store Sass configuration options and share them with other developers

This tells compilers not to create a unique CSS file based on this Sass file.

A compiler app that watches a folder full of Sass and automatically generates CSS will create `style.css` based on `style.scss`, but it won't create `navigation.css` based on `_navigation.scss`. To compile all the partials you've just created, you'll `@import` them in your main Sass file.

Each time you copy a block of code to a partial, replace it in the primary `.scss` file with `@import 'partial-name';`. If you move your footer styles to `_footer.scss`, add `@import 'footer';`. Don't include the `_` or the `.scss` in the import: just the name of the file.

If you want to get really organised and break header navigation and footer navigation into two separate partials in a `/navigation/` folder, include the folder name in the import:

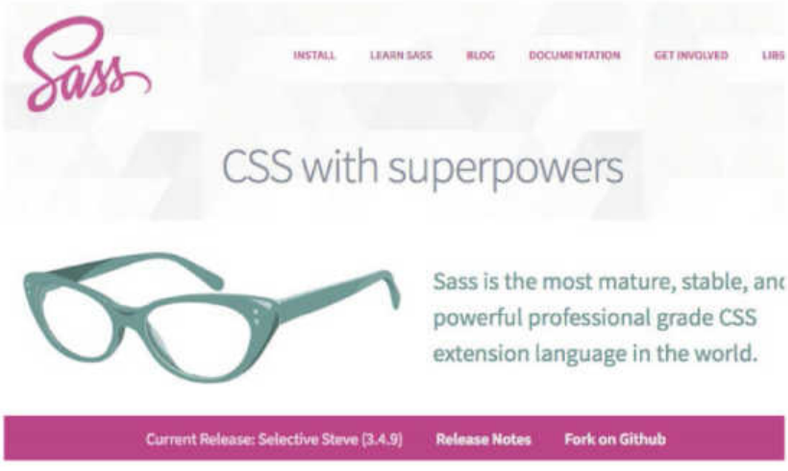
```
@import 'navigation/header-nav'; // imports /navigation/_header-nav.scss
@import 'navigation/footer-nav'; // imports /navigation/_footer-nav.scss
```

It's important to remember that the CSS cascade still applies to code written in Sass. Styles written in partials imported later have the ability to override styles in partials imported earlier.

It's also wise to import partials that contain your mixins and variables at the beginning of your primary Sass file, so that later partials can actually use those variables and mixins.

WORDPRESS COMMENTS

According to WordPress style.css requirements, we need to make sure our compiler preserves the WordPress comments at the top of style.css. When Sass' `output_style` is set to `:compressed`, it strips all comments when it compiles CSS.



Superstar Sass is the most mature, stable and powerful professional-grade CSS extension language

To ensure that those comments are left intact, add an exclamation mark (`!`) to the beginning of the comment block in `style.scss`:

```
/*!
Theme Name: Sassy WordPress Theme
Theme URI: http://jamessteinbach.com/sass/
Author: James Steinbach
Author URI: http://jamessteinbach.com
Description: From CSS to Sass Sample Theme Code
*/
// Import all your .scss partials below
```

REFACTOR CSS TO SASS

Now that we've broken a long stylesheet into smaller modular partials, we can start to refactor the original CSS to fit our Sass preferences. Some helpful Sass tools for refactoring code are variables, nesting, functions and mixins.

Now we've broken up the stylesheet, we can refactor the CSS to fit our Sass preferences

If you want to change some colours or set up a standard type scale, variables are the best way to save all that data in a single place and make site-wide changes easily. If you haven't already created a partial called `_variables.scss`, I recommend doing that now – and importing it at the top of your main Sass file. Here are some variables you may want to put in that partial:

```
$red: #FF4136;
$blue: #0074D9;
$blue-dark: #001F3F;
$orange: #FF851B
$type-small: 12px;
$type-medium: 16px;
$type-large: 21px;
$type-hero: 44px;
```

Once you've set up those variables, you can search your partials and replace values with variable names:

```
body {
  color: $blue-dark;
}
.page-title {
  font-size: $type-large;
}
```

You can use a Sass feature called 'nesting' to help make complex selectors more readable. Your starting CSS may include styles for multiple elements in the site sidebar:

```
.sidebar h1 {
  //styles
}
.sidebar p {
  //styles
}
.sidebar ul {
  //styles
}
```

You can nest styles inside of other style blocks and Sass will combine selectors to create the complex selectors. The code below will compile to the same output as the original CSS (as shown in the code above).

```
.sidebar {
  h1 {
    //styles
  }
  p {
    //styles
  }
  ul {
    //styles
  }
}
```

In nesting `&` can be used as a placeholder for the entire string of selectors above it. As nesting puts a space between selectors, it can be helpful when using pseudo-classes and pseudo-elements:

```
a {
  color: $blue;
  &:hover {
    color: $blue-dark;
  }
}
.clearfix {
  &::before,
  &::after {
    content: "";
    display: table;
    clear: both;
  }
}
```

The `&` can also be used to duplicate or reverse the order of selectors:

```
p {
  & + & {
    margin-top: 1em;
  }
}
.menu-link {
  color: $gray;
  .menu-item:hover & {
    color: $gray-light;
  }
}
```

You may be wondering if it's worth your time to refactor by nesting properties. While nesting may increase readability (although this is subjective), it's also a tool to be used carefully. Most Sass experts recommend an 'Inception rule' for Sass nesting: never nest more than three levels deep. It's wise to use Sass nesting sparingly. If it doesn't make sense in your workflow, don't force it.

If you're repeatedly calculating certain properties in your CSS, you can replace that process with a Sass function. A function takes the parameters you give it and returns a value. It won't generate CSS property-value pairs, but it can generate values for you.

Here's an example of a function to calculate faded colours on hover:

```
// This goes in _functions.scss:
@function hover-color($color) {
  @return lighten($color, 10%);
}
// This is how you use the function in other partials:
.button {
  background-color: $red;
  &:hover {
    background-color: hover-color($red);
  }
}
```

We can also refactor our code by taking repeated blocks of code and replacing them with mixins. A great example of a useful mixin is a clearfix.

```
// This goes in _mixins.scss:
@mixin clearfix() {
  &::after {
    content: "";
    display: table;
    clear: both;
  }
}
// This is how you use the mixin in other partials:
.content-container {
```

VIDEO

Micah Godbolt's 'Get Your Sass in Line' presentation covers how to share Sass files in an organised way
netm.ag/godbolt-265

RESOURCES

FURTHER READING

Here's a run-down of some good places to look if you want to explore further. The first several articles are specific to WordPress developers getting started with Sass. The rest of the links offer broader insight into good Sass organisation and maintenance processes.

'Sass for WordPress developers' by James Steinbach (*netm.ag/steinbach-265*) – This article introduces WordPress developer to Sass and recommends some useful tools

'Compass compiling and WordPress themes' by Chris Coyier (*netm.ag/coyier-265*) – Discusses methods for using Sass to generate a WordPress-compliant stylesheet

'How to use Sass with WordPress' by Andy Leverenz (*netm.ag/leverenz-265*) – Walks users through how to set up a local WordPress installation, install Sass and Compass, and start compiling Sass

'Introduction to Sass for new WordPress theme designers' by WPBeginner (*netm.ag/WPBeginner-265*) – Shows how to use the Koala app to compile Sass in a WordPress theme

'A WordPress development process using Sass and Compass' by Yanko Dimitrov (*netm.ag/dimitrov-265*) – How to set up folders and file for Compass-compiled Sass

'Architecture for a Sass project' by Hugo Giraudel (*netm.ag/giraudel-265*) – Some great ways to organise files in a Sass project

'Sass partials and the problems surrounding them' by Tim Whitney (*netm.ag/whitney-265*) – Tim explains the need to organise Sass files well

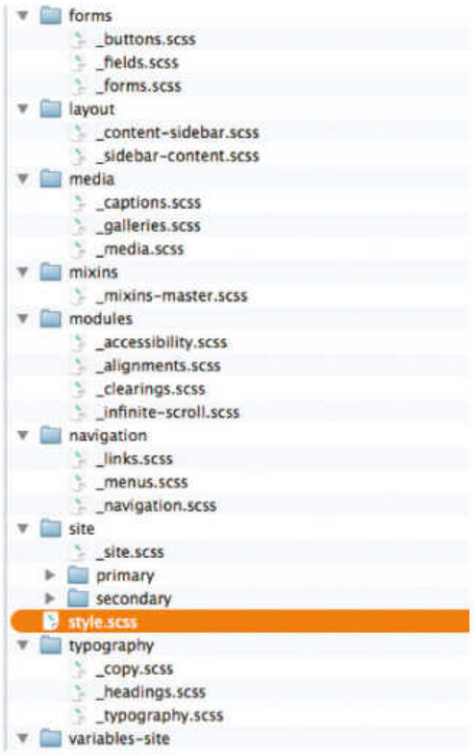
'How Evernote handles their Sass architecture' by Ryan Burgess (*netm.ag/burgess-265*) – How a large company like Evernote manages its Sass structure

```
  @include clearfix;
}
```

Mixins can also take parameters to generate customised output. This is very useful for design patterns like buttons or alerts:

```
// This goes in _mixins.scss:
@mixin alert($color) {
  border-radius: .5em;
  box-shadow: 0 0 .25em 0 rgba(0,0,0,.5);
  border-top: 4px solid $color;
  color: $color;
}
// This is how you use this mixin:
.alert-error {
  @include alert($red);
}
.alert-success {
  @include alert($green);
}
```

One common piece of advice that you may see online is to use mixins for cross-browser prefixing. I usually recommend against this, however. I find that Autoprefixer (*netm.ag/prefix-265*) is a much better way to automate prefixes. If you're unable to run Autoprefixer, and have to rely on Sass mixins,



Sorted partials The underscores.me theme sorts partials into folders including 'elements', 'forms', 'layout' and 'media'

Organising your CSS With Sass, developers can use tools like variables, nesting, functions and mixins to organise and automate their CSS

VIDEO

In this 'Getting Sassy with WordPress' presentation at WordCamp NYC, Tracy Rotton explains how Sass helps WP developers and gives an overview of some of Sass' powerful features

netm.ag/rotton-265

Compass' mixin library (*netm.ag/support-265*) allows user configuration and stays up-to-date with CanIUse data (*caniuse.com*).

ORGANISE YOUR PARTIALS

To recap, we've taken the theme's existing stylesheet apart and refactored some code to make things cleaner and Sassier. Now we can organise our partials to improve maintainability in the long run.

Remember that the cascade still matters. Sass-compiled CSS is just like plain CSS in that styles that appear later in the stylesheet can override styles that appear earlier. As a rule, import your general styles before you import specific styles.

Remember that the cascade still matters. Import your general styles before you import specific styles

Similar partials can be organised in folders. There are two ways to import Sass partials from directories. The first is to import each file into `style.scss` including the folder name in the import directive, like this:

```
@import 'base/variables';
// imports _variables.scss from the /base/ directory
@import 'base/mixins';
// imports mixins.scss from the /base/ directory
@import 'base/grid';
// imports _grid.scss from the /base/ directory
```

The second (and admittedly more complicated) method is to create a placeholder partial in each directory that imports the other partials in that directory (next column):

```
// in style.scss
@import 'base/base';

// in /base/_base.scss
@import 'variables';
// imports _variables.scss from the /base/ directory
@import 'mixins';
// imports mixins.scss from the /base/ directory
@import 'grid';
// imports _grid.scss from the /base/ directory
```

Both of these methods import the same partials in the same order. The first method is simpler on the surface, but the second method has the advantage of keeping `style.scss` neat and moving any complexity into the modules it relies on.

There are almost as many ways to organise Sass partials as there are developers trying to organise Sass partials. You'll find several good options in the 'Resources' boxout on page 112.

Here's one fairly simple organisation scheme you could use:

- `/base/` (variables, mixins, reset, typography)
- `/layout/` (grid, header, footer)
- `/vendors/` (plugins, vendors, etc)
- `/components/` (buttons, menus, forms, widgets)
- `/pages/` (home, landing page, portfolio)

SUMMING UP

This article only begins to explore the potential of using Sass in WordPress theme development. Chances are strong that you're eager for more information now, especially if this is the first thing you've read on the subject.

Check out the 'Resources' boxout for more reading around the subject – these articles include several variations on a Sass-WordPress workflow. Some of their advice differs from what I've recommended (especially on the topic of organising partials), but that's fine – find the techniques and workflows that work for you!

ABOUT THE AUTHOR

KIM CRAWLEY

t: @kim_crawley

job: Information security researcher

areas of expertise: Malware, web development, secure software development, hacker culture, social engineering, women in tech, video games

SECURITY

12 WAYS TO SECURE YOUR WORDPRESS SITE

Some simple housekeeping can help prevent your site from getting hacked. **Kim Crawley** shares her tricks for securing your sites

Developed with PHP and powered by mySQL databases, WordPress is the most popular content management system on the web. Web-delivered malware and website cracking are becoming increasingly common, and with such a large percentage of web content using WordPress as a CMS, any security vulnerabilities in WordPress' coding or framework have the potential to affect millions of websites.

In this article I am going to look at some simple measures you can take to protect your WordPress sites from malware and cracking, without needing an in-depth knowledge of security.

01 AUDIT YOUR OVERALL WORKSTATION SECURITY

First of all, make sure any and all PCs and web servers you use are kept properly secure. Make sure you're running the most recent release of your favourite web browser, and make sure that it's set to automatically patch. Do the same with your antivirus software and operating systems.

Ensure all authentication vectors you use have secure passwords, which are changed every so often. Scan your PCs and servers for malware frequently. Make sure you use proper firewalls – at the OS level, at the router level and at the ISP level, if at all possible.

Any security holes outside of WordPress, in software and hardware you use with it, can affect the CMS itself. It'd be sad to create a really secure password for your WordPress admin account, only to find out a keystroke logger had defeated all of your efforts!

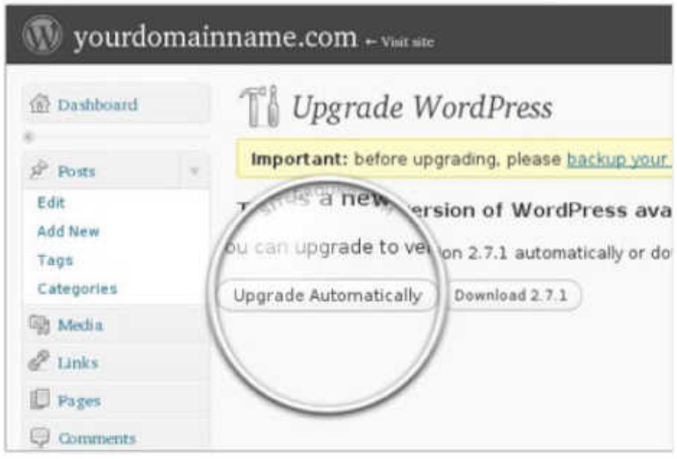
02 KEEP WORDPRESS UPDATED

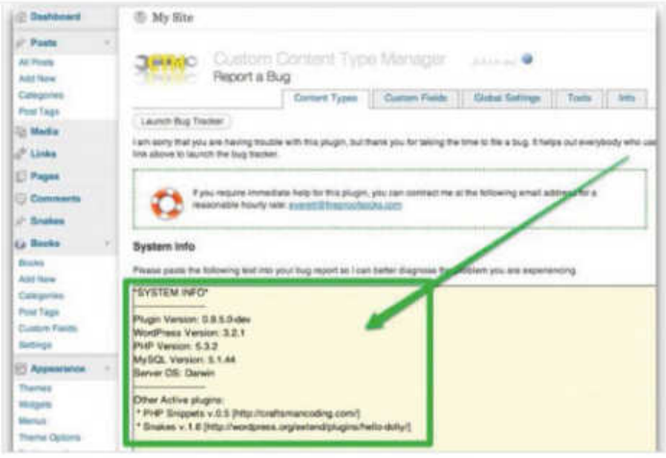
The next step is to make sure you always have the most recent version of WordPress installed. Updating WordPress is relatively quick and easy, and can be done through the WordPress panel in your web browser – so there are no real excuses.

If the most recent version of WordPress is incompatible with the versions of PHP and mySQL installed in your web server or web host, I strongly recommend you go through the effort of upgrading those to ensure your version of WordPress is up to date. It will be worth the extra time. Obsolete versions of WordPress will no longer get security patches, much the same way that older operating systems see support expiring.

03 REPORT BUGS AND VULNERABILITIES

If you ever discover security vulnerabilities on your own, do the community a favour by sending a detailed email to security@wordpress.org. If the vulnerability is in a plugin instead, email





Left Make sure you keep WordPress updated

Right Reporting bugs and vulnerabilities can benefit both you and the WordPress community

plugins@wordpress.org. You would want other web developers to report loopholes that may affect your website, so treat others as you would like to be treated! Just avoid writing about those newly discovered vulnerabilities on the web or on social networking sites, so that information doesn't fall into the wrong hands.

04 CHECK FOR EXPLOITS

Every so often, run the Exploit Scanner plugin to check for indications of malicious activity. Exploit Scanner doesn't directly repair any issues, but it will leave you a detailed log to troubleshoot with. If you ever suspect cracking, that's the time to run that plugin, as well.

It's much easier to crack into a WordPress site when you know which version is installed

05 DISABLE CUSTOM HTML WHERE POSSIBLE

WordPress can use custom HTML for various functions. If that isn't absolutely necessary for the form and function of your website, you may want to disable unfiltered HTML by adding the following to your 'wp-config.php' file:

```
define( 'DISALLOW_UNFILTERED_HTML', true );
```

06 DON'T LOOK BRAND NEW

Remove all default posts and comments. If malicious hackers find those on your site, it may indicate to them you have a new WordPress site, and brand new sites are often easier to crack into.

It's also easier to crack into a WordPress site when you know which version is installed, so be sure to hide this information. This is done in two places.

The first is the meta generator tag in your template. That's found in `wp-content/{name` of your WordPress `theme}/header.php`. Look for something like `""` and remove it.

The other element is in your RSS feed. Open up 'wp-includes/general-template.php' and look around line 1858. Find:

```
function the_generator( $type ) {
echo apply_filters('the_generator', get_the_
generator($type), $type) . "\n";
}
```

Make sure a hash is applied next to the "echo" command so that it looks like this:

```
function the_generator( $type ) {
#echo apply_filters('the_generator', get_the_
generator($type), $type) . "\n";
}
```

Also, remove all instances of 'Powered by WordPress' footers, as crackers use the phrase to find sites to crack into via search engines. That footer also indicates new WordPress sites, or sites developed by newbies, whether or not that applies to you.

It's a good idea to delete '/wp-admin/install.php' and '/wp-admin/upgrade.php' after every WordPress installation or upgrade. After all, those particular scripts are only ever used during the installation and upgrade processes, and aren't required in the everyday development of your site. Don't worry though – you can still, of course, upgrade WordPress without those files, as all upgrades contain those scripts.

Next, change a couple of the file and directory name defaults. Go to Settings > Miscellaneous ▶

in your admin console and change the names of 'wp-content/directory' and 'wp-comments-post.php'. Make sure you change the template URL within the template and 'wp-comments-post.php' accordingly, in order to maintain the function of your site.

07 HIDE INDEXES

Make sure you disable public access to indexes whenever possible. If people can find the files in your site's 'wp-content/plugins/' directory without being authenticated, it's a lot easier to crack into your site through plugin vulnerabilities.

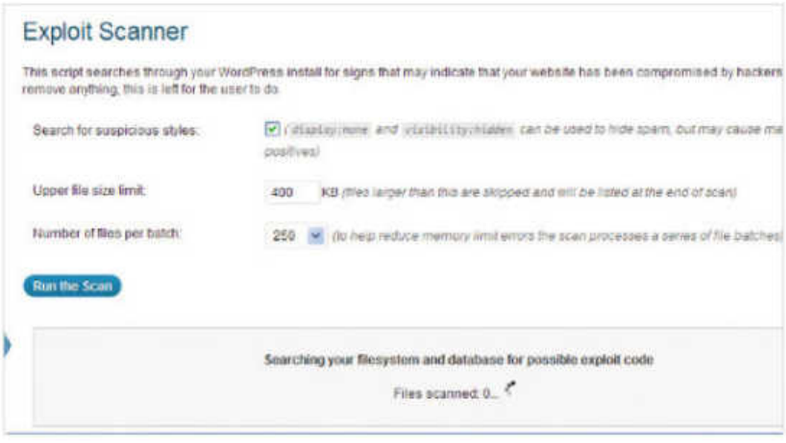
If your web server runs Apache or another OS that uses '.htaccess' files, it's simple to do. Find the '.htaccess' configuration file in your site's main directory. That's the directory that contains 'index.php'. Insert the text `Options -Indexes` anywhere in the file.

Alternatively, if you can't alter a '.htaccess' file, upload an 'index.html' file into your main directory. You could make that web page have a similar look to your site's PHP web pages and insert a hyperlink to your 'index.php' file if you'd like. But obviously, in a site that uses WordPress as a CMS, visitors won't see your 'index.html' file unless they type a specific path to it in their web browser address bar. Alternatively, you could make your 'index.html' file a 0-byte placeholder.

In case your web server ever has problems computing PHP files, it's crucial to block directories that are only accessed by your server. If the PHP source code is ever displayed in a visitor's web browser rather than the web page it's supposed to render, they may find database credentials or in-depth information about the PHP/mySQL programming of your site.

Your site's wp-includes/ directory is the most important one to block. Find the '.htaccess' file there and insert:

Malicious activity
Running Exploit Scanner every will warn you of any indications of malicious activity




```
RewriteRule ^(wp-includes)\/.*$ ./ [NC,R=301,L]
```

If there are or will be subdirectories of wp-includes/, insert the following code for each one in the same .htaccess configuration file:

```
RewriteRule ^(wp-includes|subdirectory-name-here)\/.*$ ./ [NC,R=301,L]
```

08 BACK IT UP!

WP-DB Manager is excellent for backing up your entire WordPress site, but it'll also alert you to mySQL vulnerabilities and let you know when parts of your database are publicly accessible.

Always be sure to properly back up the content of your site. In a worst-case scenario, at least keeping backups will allow you to easily restore your site. With WP-DB Manager, you could also use Online Backup for WordPress. The backup the plugin creates can be stored in your email inbox or on your PC, or you can use the 100MB of free storage space on developer Backup Technology's own secure servers.

09 INSTALL SECURITY PLUGINS

As well as the Exploit Scanner plugin (see tip four) which you should run on your site every so often to check for vulnerabilities and cracking attempts, there are a number of other WordPress plugins I recommend you install and use. When used properly, they can harden your WordPress site effectively.

With Exploit Scanner, you can also use WP Security Scan. Not only will the plugin look for vulnerabilities, but it'll also give you specific advice on how to block them.

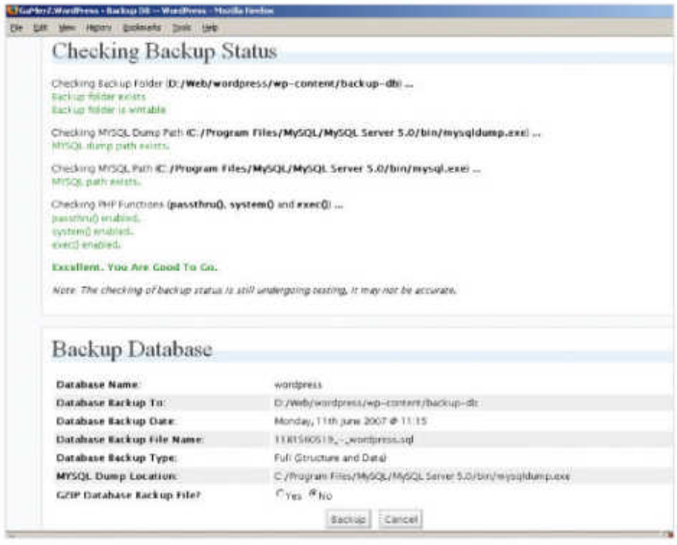
To prevent cracks to find your login credentials, be sure to encrypt your login packets with Login Encryption. That plugin uses both DEA and RSA algorithms for enhanced security.

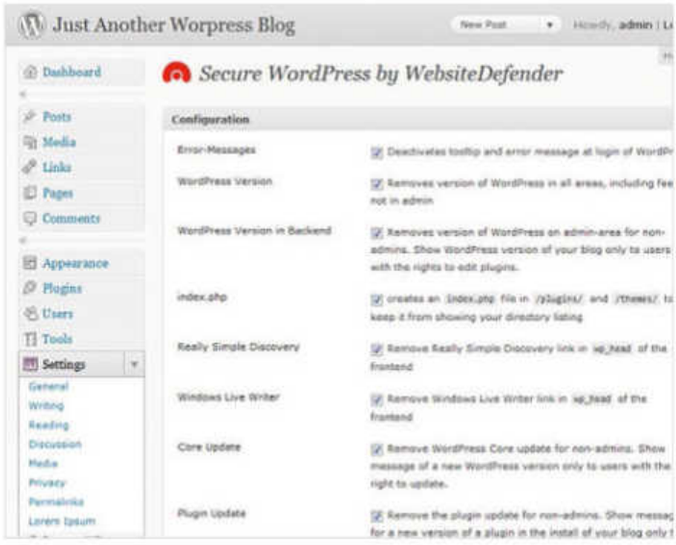
10 INSTALL PLUGINS FROM THE ADMIN PANEL

Configure the Limit Login Attempts plugin to prevent brute-force attacks. With the plugin, you can set a maximum number of login attempts, and also set the duration of lockouts in-between.

The User Locker plugin works in a similar way. With it, you can set a maximum number of invalid authentication attempts before the account is locked. There's also an excellent plugin for securing your entire admin panel. Try the Admin SSL Secure Plugin to encrypt your panel with SSL.

Another strong plugin for securing your site's login is Chap Secure Login. By using this, all of your login credentials (except for usernames) will be encrypted with the Chap protocol and SHA-256 algorithm.





Left Whatever you do, make sure you back up

Right Installing useful plugins can help protect your site further

As mentioned before, it's an excellent idea to change as many WordPress defaults as possible. With Stealth Login, you can create custom URLs for logging in and out of your site. Block Bad Queries will try to block malicious queries made to your site. It looks for `eval(` or `"base64"` in request URIs, and also for request strings that are suspiciously long.

An anti-malware shield can be applied to your entire site with the AntiVirus plugin, as well. This

Your WordPress site should be on a schedule for vulnerability and malware scanning

will look for viruses, worms, rootkits, and other forms of malware. Be sure to keep it updated!

Finally, remember: when you choose and install plugins on your site, also be sure to only install plugins offered through your admin panel or under the plugin directory at *WordPress.org*. Outside plugins may be secure, but it's best to mitigate the risk. Officially released plugins are audited for security and scanned for malware.

11 INSTALL OTHER USEFUL PLUGINS

WordPress sites are frequently targeted by spambots. You can spend a lot of time going through comments on your site, and the majority of your pending comments may have to be marked as spam. Imagine what those spambots can do to your site, beyond giving you a lot of work! For that reason, I recommend installing Bad Behavior (*netm.ag/bad-271*). By logging your site's HTTP requests, you can better troubleshoot spambot issues. Furthermore, the plugin will limit access to your site when a bot hits it.

With Bad Behavior, you can also use User Spam Remover. It will remove unused user accounts on your site. You can set an age threshold to those settings and you can also configure a whitelist.

12 AND FINALLY ... PUT EVERYTHING TOGETHER

Keeping your WordPress site hardened for security is an ongoing responsibility, just like all other areas of IT and development security. You can't just configure a number of settings or programs and then forget about it. Your WordPress site should certainly be on a schedule for malware and vulnerability scanning, and logs should be kept and analysed.

By keeping your WordPress site secure, you're doing your part to prevent malicious activity that could not only harm websites, but also web servers and users' PCs, tablets and smartphone devices. Think about it!

As WordPress is such a common CMS on the web, knowledge about the design and configuration of the console is readily available, and certain hacks could work on perhaps millions of websites. Fortunately, knowledge about WordPress security is abundant, for much the same reasons. In the ongoing maintenance of your website and web server, always be security-minded. You can then have proper control over your web content, and do your part to make the internet a better place.

ABOUT THE AUTHOR

RYAN MCCUE

w: *rmccue.io*

t: @rmccue

job: Senior engineer, Human Made

areas of expertise: PHP, WordPress, JavaScript, HTTP, APIs

REST

BOOST YOUR WORKFLOW WITH THE REST API

Ryan McCue explains how you can use this new API for pulling content and data out of WordPress, as well as writing or updating posts

Over the past 12 years, WordPress has grown from being a small side project to a tour de force. Recent studies estimate almost a quarter of all visible sites on the web are powered by it. However, while WordPress has been growing, websites have evolved.

We've seen the rise of web applications that take simple HTML pages and enhance them with complex JavaScript and API interactions. This has taken sites and software far past mere publishing to something far more interactive, using sites as full platforms. In the meantime, WordPress has been chugging away building traditional sites, but it hasn't had a standard option for those who need this functionality. At least, not until now.

For the past two years, a small team of core contributors has been working on a solution to this: the REST API (*wp-api.org*). The project was started at the end of 2012 as a side project, then became an official Google Summer of Code project for WordPress in the middle of 2013. Since then, the plugin has been under active development

VIDEO

Rachel Baker has created an in-depth video tutorial to help you get started with the REST API. Purchase the recording at *netm.ag/tutorial-271*

Demo A demo of the API with a console is available at *demo.wp-api.org*

under the Feature as a Plugin core development process, intended to be merged into WordPress core. Despite being in a plugin, the project is treated as a future feature, with easy installation built in to beta versions for WordPress.

The REST API has already seen plenty of use by production websites. *Wired.com*'s recent redesign included merging all of its sub-sites into one, and building a REST API-powered site on top of it, without using WordPress' built-in theming. The API is also a key component of the New York Times' live-blogging platform, which uses both the REST API and the companion JavaScript library.

'Representational State Transfer' is a set of best practices and standard uses for HTTP requests

GETTING SET UP

Although at time of writing the API is yet to be finalised, we can start building with it today. As it is just a regular plugin for WordPress, you can follow the same installation process that you would with normal plugins.

To work with version 2 of the API, you'll need to download the plugin from GitHub (*github.com/WP-API/WP-API*). For the more advanced creation and update steps, you'll need the Basic Auth plugin (*github.com/WP-API/Basic-Auth*). You'll also want to make sure you have pretty permalinks on – although it's possible to use the API without them, it makes things a bit more complicated.

Once you've activated the plugin, the API will be available at */wp-json/* on your site. For example, if your site is at *http://example.com/*, the API would be

FOCUS ON

WHAT EXACTLY IS REST?

REST stands for Representational State Transfer and was originally conceived by Roy T. Fielding in 2000. It is a set of best practices and standard uses for HTTP requests. At the core of the concept is 'resources', which are the real objects in your system. Resources are represented by URLs, which act as the name to refer to the resource. These resources have four basic operations (or verbs) you can perform: reading with `GET`, creating with `POST`, updating with `PUT` and deleting with `DELETE`.

You've probably seen a few of these before when working with forms, but REST takes the concept a little further by matching these available HTTP verbs with operations on the resource. These resources can be anything from posts or pages in the WordPress world, through to representations of the site. While not everything you might want to do maps exactly to resources and verbs, the concept matches up nicely to the vast majority of use cases.

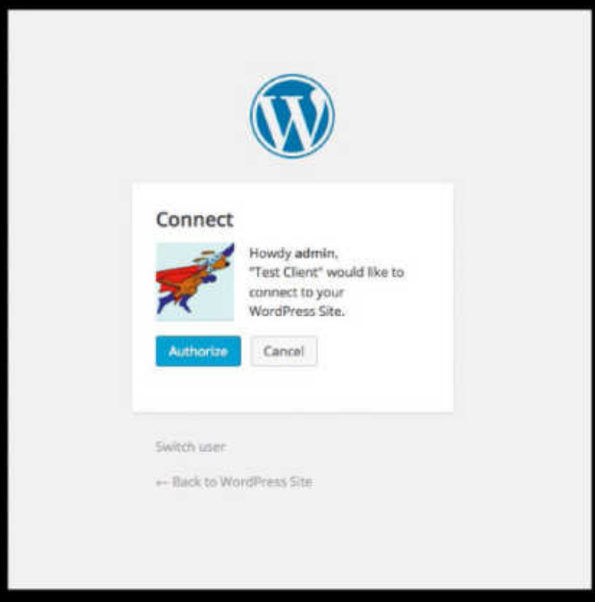
FOCUS ON

AUTHENTICATION

We're using Basic authentication here to simplify the process of authenticating while learning, but it's not the best choice for more serious work. Basic authentication requires you to give your username and password to the application, which means if the application is compromised or you don't want to use it any more, there's no way to revoke access aside from changing your password completely.

OAuth is designed as a solution to this. Through a semi-automated process, applications request access (via your user), and the site gives them a token to access the site. If the application is compromised, you can simply revoke the token. This is a more complex process, as it needs two-way communication between the site and the application. As WordPress also needs to work across all sites, OAuth 1.0a has to be used, which is an older version of the specification that works on non-SSL sites. You can download the OAuth plugin from *github.com/WP-API/OAuth1*.

For plugins and themes using the site directly, you can instead piggyback off WordPress' cookie-based authentication. This works the same way as normal Ajax requests in WordPress code, allowing enhancement of plugins and themes with API data. You'll need to pass a special `"nonce"` parameter as a Cross-Site Request Forgery token with the request, which can be generated by calling `wp_create_nonce( 'wp_rest' )`. There's also a JavaScript API bundled with the plugin, which handles this automatically.



OAuth link OAuth is done via a Link page on your site

available from *http://example.com/wp-json/*. If your site is at *http://example.org/foo/*, the API would be available from *http://example.org/foo/wp-json/*. This is called the API root, and we'll be using it a lot, so remember it.

For viewing in your browser, you can install JSONView (*jsonview.com*), available for both Firefox and Chrome. This will let you view the available data and click around through the links in the API to find everything that's available.

For more advanced usage, Postman for Chrome (*getpostman.com*) or Paw for Mac (*luckymarmot.com/paw*) are fully featured API clients that will enable you to work with the API in a more complex way. For now, I'm going to be using JSONView for simply viewing data.

Now we have the API installed, we're ready to start looking at the data. At the API root you'll find the index. This gives you a bit of information about the site and the API itself, including available authentication methods and plugin APIs. You'll also notice a list of available 'routes'. Routes are the URLs you use to access the various objects and collections (lists of objects) via the API. This includes all the core objects in WordPress: posts, comments, terms and users.

Let's take a look at the posts route. In the index, you'll see a route called */wp/v2/posts*. This route is a collection, which lists a bunch of objects and contains links to the individual items. The */wp/v2/posts* route gives you access to all the posts available on the site. It also exposes the ability to filter this down to search and browse through all the post data on the site.

This gives you the flexibility of `WP_Query` through the API, including text search, filtering by

Viewing data JSONView can be installed in your browser to make viewing JSON data a bit easier

author, and querying by tags and categories. Click through to the posts route or head to */wp/v2/*posts after your API root (for example, something like '*example.com/wp-json/wp/v2/posts*') and you should see something like this:

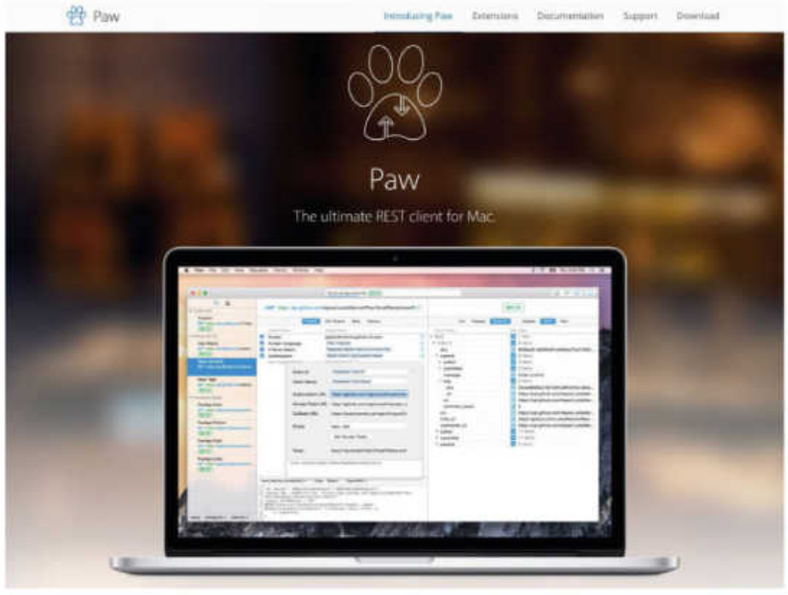
```
[
    {
        "id": 89,
        "date": "2015-06-26T02:44:23",
        "title": {
            "rendered": "Hello world!",
        },
        "content": {
            "rendered": "<p>Welcome to WordPress. ...</
p>"
        },
        ...
    }
    {
        "id": 1,
        "date": "2015-06-03T22:12:40",
        ...
    }
]
```

Routes are the URLs you use to access the various objects and collections via the API

This is a JSON list (denoted by square brackets, and with items separated by commas) of the various posts you have on your site. By default, the number of items is limited to 10. Each post is a JSON object (denoted by curly brackets) giving data about the post and fields available on it.

The data here is basically everything that WordPress knows in relation to the object. For posts, this includes normal fields like the post title and content, publish date and status, through to custom meta and tags attached to the post. You'll also see links to the single post, which look like */wp/v2/posts/42*, where 42 is the post ID.

All the data returned by the API follows the same pattern. WordPress objects are represented by JSON objects. You can access these directly via the single link (which looks like */wp/v2/<noun>/<id>)*, or you can access the collection they're part of (this looks like */wp/v2/<noun>*). The single link is always an object, and the collection is always a list of objects.



Paw client Paw is a full-featured standalone REST API client. It includes code generation for many languages

WRITE THE WEB

The REST API isn't just for pulling content and data out of WordPress. It can also be used for writing new posts, or updating existing ones. But don't be afraid, it's actually super easy!

The input format for data matches the output format, so to update a post you can simply send back the exact data you just received, but with any fields you want changed filled with new data. Or, if you want to keep it simple, you can send back just the changed fields. This data is sent as a JSON object, just like you received it.

Before, when we wanted to get data for an object, we accessed it in our browser. Behind the scenes, this involved sending off a `GET` request via HTTP that asked the server to get the data for the given URL (resource). With REST, we use `POST` for creating, which will be familiar to you if you've ever written a HTML form. For updating, we use `PUT`, which works in a similar way to `POST`.

To start off, we'll create a new post. To do this, we want to send a `POST` request off to */wp/v2/posts* on our site with the following data:

```
{
    "title": "Hello!",
    "content": "Hi, this is my new post."
}
```

To actually create this, we'll need to send off authentication too. The easiest way to do this is to use HTTP Basic authentication, which passes your username and password in plain text. This isn't ▶

VIDEO

For a 30-minute introduction to using the API, with a walkthrough on how to extend the API further with your own content, go to *netm.ag/walkthrough-271*

the most secure or safest way to authenticate, but it's fine for development (for more on this, see boxout on page 120). This isn't built into the API, so you'll need to make sure you have the Basic Auth plugin activated. Basic authentication uses username:password as a string, which is then Base64 encoded before passing in the header.

Here's what our request should look like with headers and our body:

```
POST /wp-json/wp/v2/posts
Host: example.com
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=

{"title":"Hello!","content":"Hi, this is my new post."}
```

When we send this to the server, we should then get back a post that looks just like the ones we received before, including a post date, automatically generated excerpt and author data. This should give you a 201 Created status code back too, plus a Location header indicating where you can access the post (this is a URL which looks like *http://example.com/wp-json/wp/v2/posts/42*).

Postman request
Postman is a REST API client designed to make interacting with APIs easy. Here we're sending the POST request to create a new post

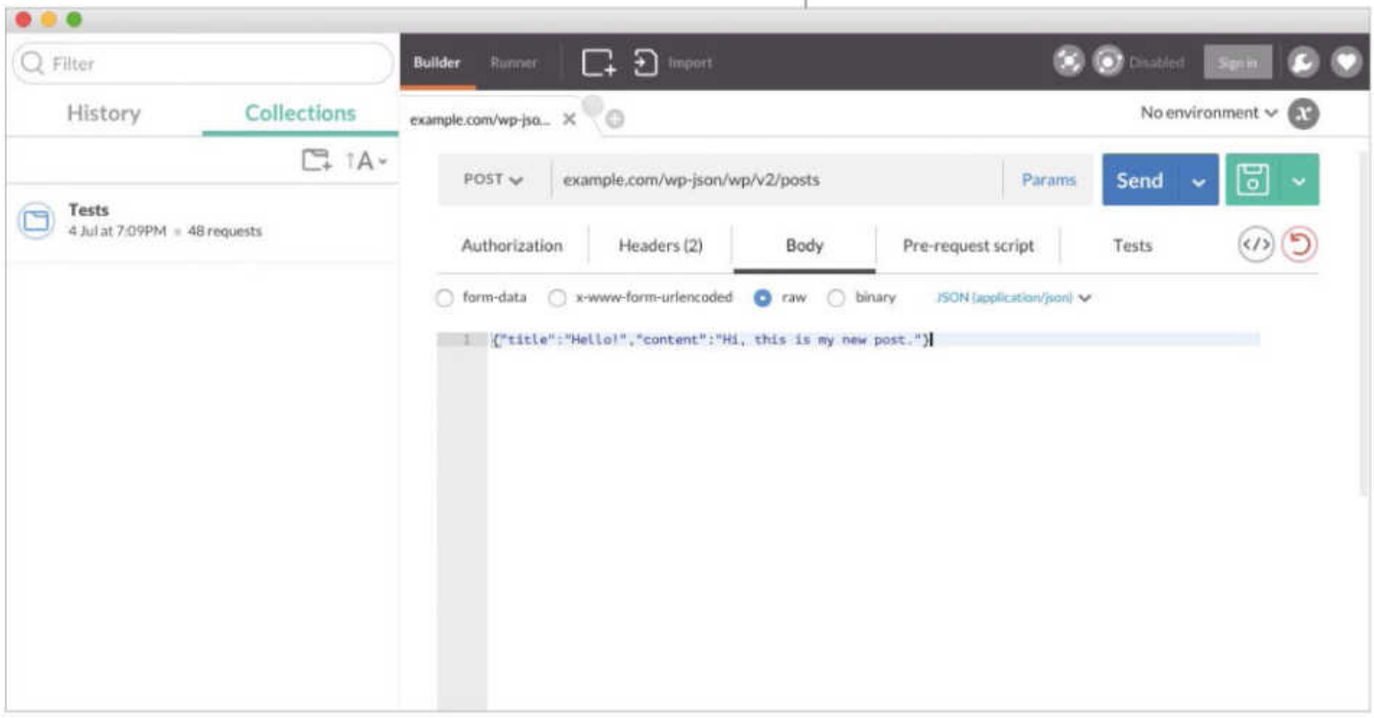
```
{
    "id": 42,
    "title": {
        "rendered": "<p>Hi, this is my new post.</p>",
    }
    "status": "draft",
    "date": "2015-05-01T04:50:00",
    ...
}
```

If you check your site admin, you should now see the post has appeared in your dashboard. Congratulations! The post has a bunch of default data attached to it, like comment status, auto generated slug and sticky status. You'll notice the post also has the draft status by default, so it won't show on your live site just yet.

Let's fix this by taking the post we received before and publishing it. To do this, we need to update the post we just got. To update items, we will send a `PUT` request to the URL we got back in the Location header. We only need to send the data we want to change, so we'll set the status property to `"publish"`. We'll also need to send authentication along with the request.

```
PUT /wp-json/wp/v2/posts/42
Host: example.com
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=

{"status":"publish"}
```

We should get back a 200 OK response from the server, indicating that the post has successfully been updated. We'll also get back the updated post



in the body, and we should see `"status": "publish"` there. Check your site, and you should see that the post is now live. Huzzah!

To finish off, let's delete the post to clean up our testing. We need to send a `DELETE` request to the same URL. We'll following almost the same process as before – we still need to send authentication, but this time we don't need to send any extra data along.

We should have a request that looks like this:

```
DELETE /wp-json/wp/v2/posts/42
Host: example.com
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=
```

We should see the post disappear off our site now. Awesome work – we just went through the full publishing process with WordPress, and all via the REST API.

BUILDING PLUGINS

The API is designed to be easily usable with WordPress-based code too. If you have plugins or themes running on a site, you can use the REST API with minimal effort. In fact, authentication is

The REST API is designed to be easily usable with WordPress based code too

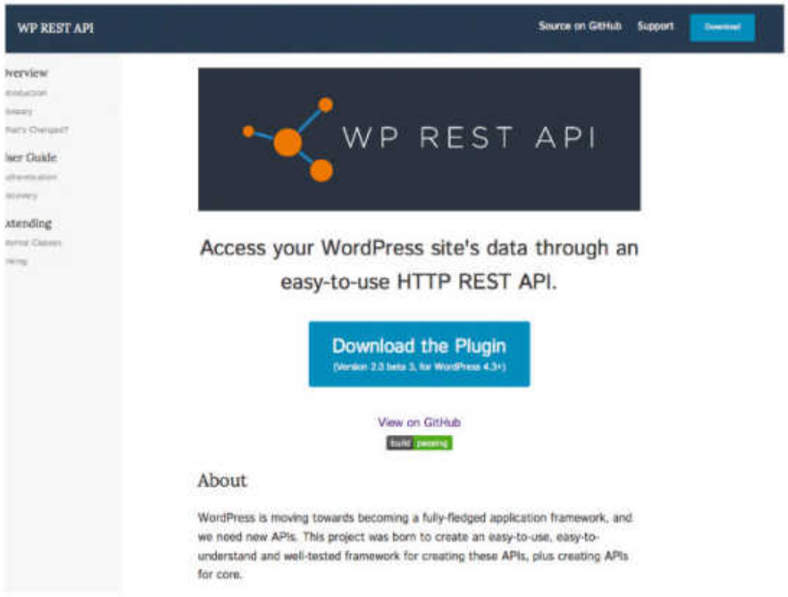
handled for you automatically via WordPress' built-in cookie authentication system – you need only to pass an extra Cross-Site Request Forgery token along with your requests.

The REST API plugin also includes a JavaScript-based API that you can use. This includes methods for accessing data from the site, as well as creating, updating and deleting this data. All of this is achieved through Backbone-powered data models and collections.

To use the API, you need to enqueue the script in your theme or plugin. In your plugin or theme, simply call `wp_enqueue_script( 'wp-api' )`. Let's run through our example from before, first accessing the post:

```
var posts = new wp.collections.Posts();
var post = posts.get( 42 );
```

To create a new post, we use similar code to before, but in JavaScript:



Docs and more Full documentation is available at *v2.wp-api.org*, including guides on authentication processes

```
var post = new wp.models.Post();
post.set( 'title', 'Hello!' );
post.set( 'content', 'Hi, this is my new post!' );
post.save();
```

We've just replicated the creation process from before, without having to worry about the intricacies of authentication or HTTP requests. We can then go and publish this as well:

```
post.set( 'status', 'publish' );
post.save();
```

NEXT STEPS

You've just learned how to interact with the API both via normal HTTP API requests and through the JavaScript API. We've barely scratched the surface of what's available through the API though. Luckily, it includes built-in documentation on possible routes and methods, as well as the data that is available and the format it takes. Take some time to browse through the API from the index at the API root, and discover what's there.

The REST API is currently aiming for integration into WordPress core by the end of 2015. Comments on the project vision, bug reports and feature requests are always welcome as we gather feedback before integration. Simply head over to GitHub and leave feedback on a new GitHub issue.

The API has the ability to change and improve how we work with WordPress. We're looking forward to seeing what you build with it now and in the future. Happy hacking!

VIDEO

For a 30-minute introduction to using the API, with a walkthrough on how to extend the API further with your own content, go to *netm.ag/walkthrough-271*

EXCHANGE

Practical advice from industry experts

FEATURING...

JOSH POLLOCK

Josh is a WordPress developer, educator and tech entrepreneur
w: *JoshPress.net*
t: @Josh412

RACHEL MCCOLLIN

WordPress specialist Rachel is a web designer, developer and writer
w: *rachelmccollin.com*
t: @rachelmccollin

TIM NASH

Tim is a consultant who specialises in WordPress development and training
w: *timnash.co.uk*
t: @tnash

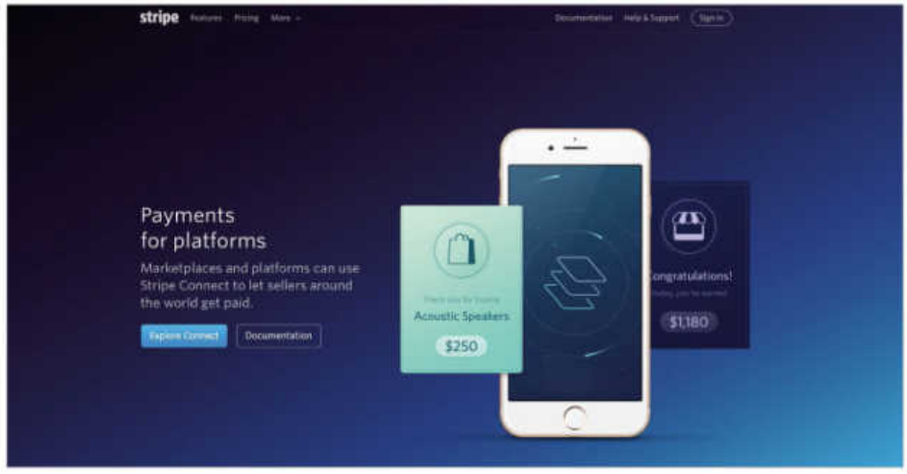
AMBER WEINBERG

Amber is a frontend developer who specialises in WordPress and responsive projects
w: *amberweinberg.com*

FEATURED QUESTION

What's the best shopping cart payment option, without redirecting to another site and if you don't want to use a credit card?

Tony Notermann, Minnesota, US



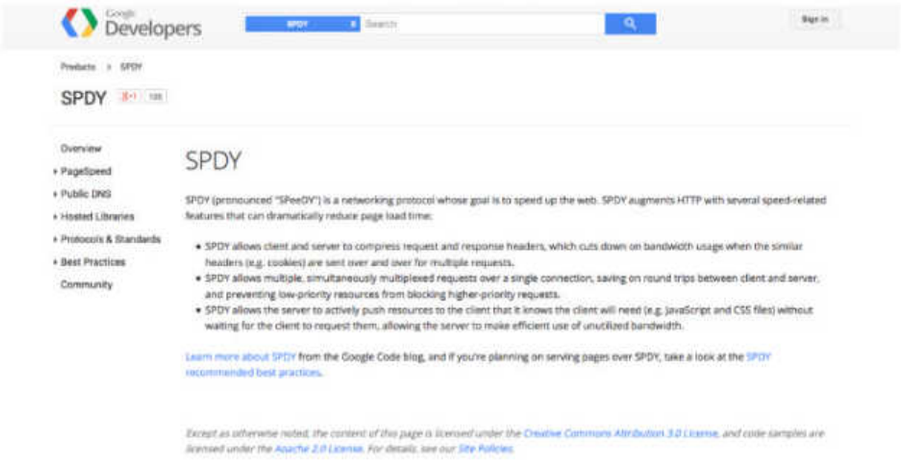
Payment options Stripe is a set of APIs and tools that will help online payment management easier

JP: My preferred way to handle payment processing is with online payment tool Stripe (*stripe.com*). Stripe can work as fields in your checkout form or in a modal. There are great implementations for all the popular ecommerce plugins like WooCommerce and Easy Digital Downloads, as well as for form plugins like Gravity Forms, Ninja Forms or Caldera Forms. PayPal Payments Advanced (*netm.ag/paypal-271*) is trickier and more expensive to set up than the regular PayPal Express Checkout, but it will work without having to go to PayPal's site.

DOMAIN STORE

Why does WordPress store the domain in the database?
Michael Lynch, Toronto, CA

RM: Sometimes you might need to tell WordPress that the URL where WordPress is located and the site domain are different – for example if you've got WordPress installed in a subdirectory but want the site to behave as if it's in the root directory. You can change the settings for the site and WordPress domains via the General Settings screen. WordPress saves this setting to the database and uses it when displaying your site to users.



SPDY is pronounced 'speedy' and is a protocol that's been designed to speed up the web.

MINIFICATION

What's the best practice for concatenating and minifying CSS/JS? Using `wp_enqueue_scripts` versus Grunt?

Matt Walters, London, UK

TN: It's not so much about best practice but what fits your workflow. Where possible, I would try to avoid leaving it to WordPress, as we don't want to put extra strain on the server at page load. So if we can pre-minify as part of our workflow, before getting to WordPress, then this is going to be a more efficient way of doing things. Grunt is a good tool for this – and we can let the web server handle it with tools like the PageSpeed Module (*netm.ag/speed-260*).

SPDY AND HTTP2

What's your current feeling on SPDY/HTTP2? Should sites (not necessarily WordPress ones) be trying to implement it?

Phil Nash, London, UK

TN: While HTTP2 is still in its infancy, it's based on the more mature SPDY protocol (*netm.ag/SPDY-260*). The latter is already being used in the production of servers on major web services such as Google, Facebook and Twitter, and in most circumstances it helps reduce page load.

If you have root access to a server, implementing SPDY is a fairly simple process. It's also supported by major vendors such as Apache and NGINX. Implementing SPDY on your server is a quick and relatively simple win for performance and security.

WORDPRESS BOTS

I've seen lots of stories about WordPress being used in DDoS attacks. Should I worry about my site becoming part of a botnet? Are there any security precautions I should be taking?

Jason Robson, Cardiff, UK

AW: Like anything else on the web, WordPress is susceptible to security issues. The best way to prevent this is to keep your core and plugins updated and use as few plugins as possible. Never use 'admin' as a username and ensure your passwords are secure. If you're still worried about attacks, there are great articles on the web that talk about moving the core files elsewhere and modifying '.htaccess' files. But usually, the basics are enough.

NON-SERVER FIXES

What's the best way to improve performance on a cheap host that you only have FTP access to?

Robert O'Rourke, Liverpool, UK

TN: With server side off-limits, look at how you can reduce http requests (minification and concatenation of CSS/JS), reduce the size of files (especially images) and look at lazy loading. On the theme side, if possible look at reducing the number of DB queries and caching fragments. You could also consider going completely static. Depending on your hosting you may wish to offload static assets to a CDN. Finally consider an alternative host – FTP access-only is never a good sign.

3 SIMPLE STEPS

What happens if I have multiple menus where some links are the same, but not all?

Michael Russell, Dublin, IE

RM: The Menu Manager in WordPress is the best way to build multiple menus with differing links.

ADD MENUS TO WIDGETS

You can add menus to widgets in your site – here I've created a Categories menu. You could code a list of categories in your theme, but using menus gives you more control over your menu's content and order.

POSITION YOUR WIDGET

Then add the Custom Menu widget to the widget area where you want to display your menu. Here I'm adding it to the footer. You can give your menu any title you want.

EDIT IN THE MENUS SCREEN

This then displays the menu in the site's footer. Whenever you want to edit that footer menu, just edit the menu you created in the Menus screen, and the widget will automatically update.

3 SIMPLE STEPS

How would I go about adding sort button functionality to my events custom post type?

Rick Yentzer, Boston, US

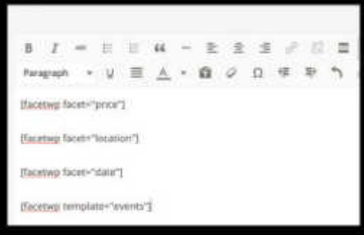
JP: FacetWP (*facetwp.com*) is the best plugin for this type of search – it lets you create search filters that update instantly on the page.

START A TEMPLATE

Once Facet is installed go to the admin page. Create a template to show posts in the 'events' post type.

ADD FACETS

Create Facets to search by custom fields, like date, price or even taxonomies – a custom taxonomy for locations would be smart here.



CREATE SHORTCODES

Create a new page and add the FacetWP shortcodes to show your template and each of your facets.

Template resources Daniel Pataki's guide will help you learn how to create child themes

TIME TO MOVE ON?

Do you think WordPress has had its time? So many devs seem to hate it.

Mustafa Kurtuldu, London, UK

TN: WordPress is currently managing around 24 per cent of websites and that figure continues to grow. The number of contributors is increasing and there's an increase in the rate of adoption in enterprise stacks. WordPress is not going anywhere. As for haters, that's just the nature of the web.

ADVANCED CUSTOM FIELDS

Why doesn't WordPress just integrate the Advanced Custom Field plugin (or something similar) from the start?

Dan Davies, Flint, UK

JP: Because not everyone needs what Advanced Custom Fields (ACF) does, and not everyone who does need it wants to use ACF. Improvements to WordPress' metadata API, which enables plugins like ACF to work, are planned.

ACF is a great plugin, but some people prefer Pods, Custom Field Suite, CMB2 or other similar tools. Some users might prefer to work without a plugin like that at all. It's not the job of WordPress core to dictate to users which one to choose.

TEMPLATE RESOURCES

What are the best resources to turn to if I want to learn how to make custom WordPress templates?

Brian N. Williams, Vancouver, CA

RM: Creating your own themes lets you design and build exactly the way you want. There are two good places to start learning: either convert existing HTML to a theme, or create a child theme for an existing theme. My series for tuts+ on converting HTML to a theme will help you learn how to do that (*netm.ag/static-271*), while Daniel Pataki's guide to creating child themes will give you everything you need to know on that topic (*netm.ag/child-271*).

JQUERY CALLS

WordPress can include an extra call to jQuery on the frontend. How do you prevent that?

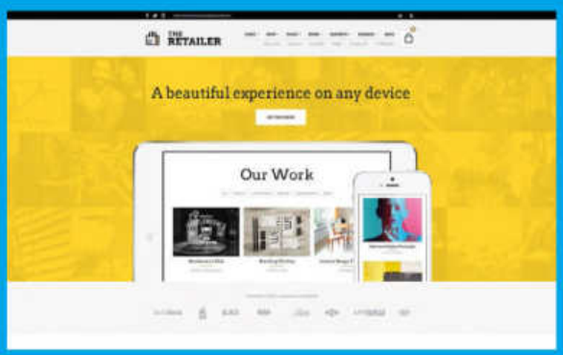
Tim Evko, New York, US

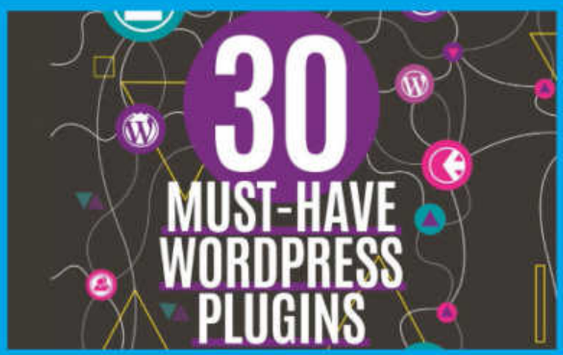
TN: If you are seeing multiple versions of jQuery loading, it's normally because a theme or plugin is loading the additional version, rather than WordPress. You can check if WordPress has jQuery registered by using `wp_script_is( 'jquery' );`. To de-register jQuery:

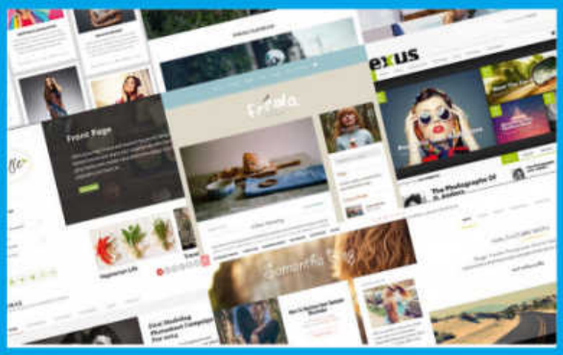
```
function net_remove_jquery() {
  if ( ! is_admin() && wp_script_is( 'jquery' ) ) {
    wp_deregister_script('jquery');
  }
}
add_action('wp_enqueue_scripts', 'net_
remove_jquery');
```

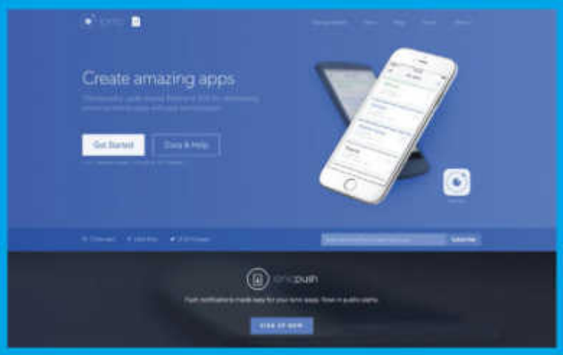
The above code checks we are on not on an admin screen and that jQuery is in our scripts list, before de-registering it. However, if a theme or plugin is forcing a version of jQuery in the head and not through registering the script with WordPress, that script will remain. Ideally you want to remove the spurious call via the plugin or theme rather than removing the WordPress version.

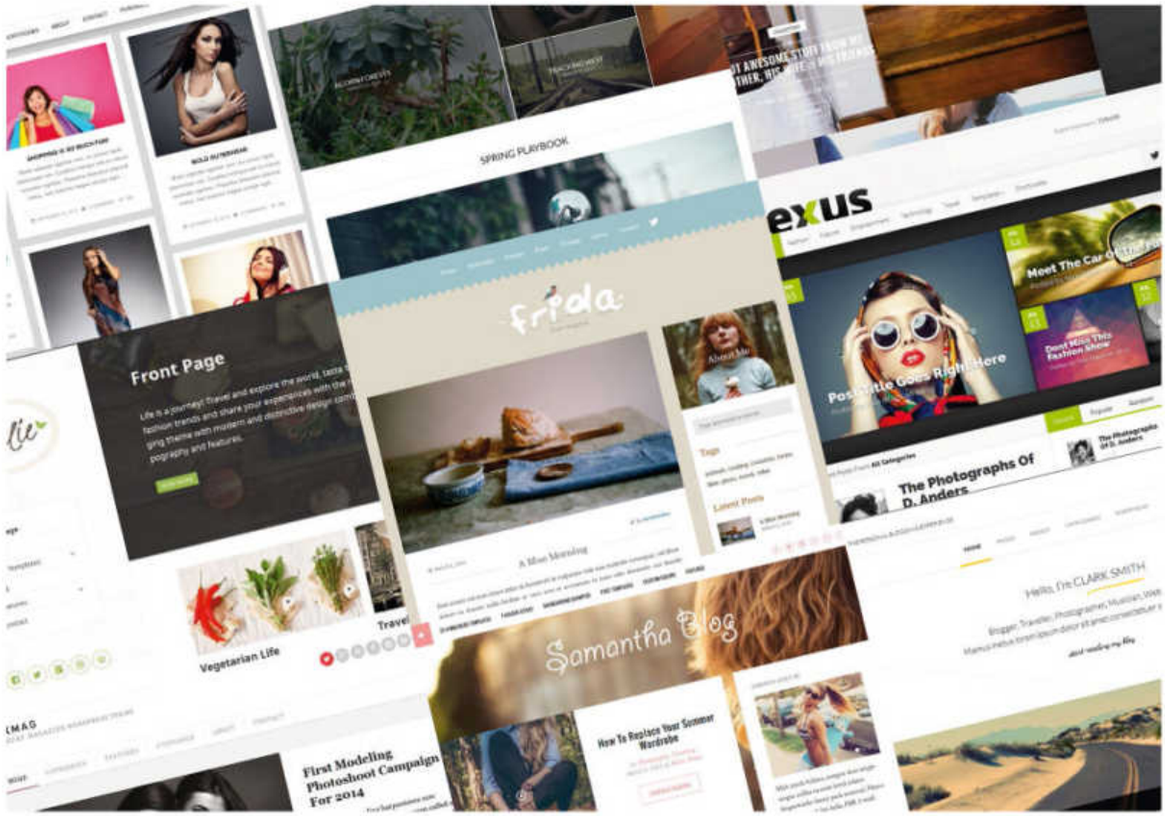
RESOURCES











THEMES

25 NEAT WORDPRESS BLOG THEMES

Mike Brown rounds up a selection of the best quality WordPress themes for you to use for your blogs

There are currently over 60 million WordPress users. The platform powers around 24 per cent of the internet, and there are more sites being added each day. To support such a colossal number of websites, bloggers and website developers need themes. Here are 25 neat WordPress themes you can use for your blogs.

1 CHOW

netm.ag/chow-271

Chow is a premium WordPress theme especially for budding food bloggers. It has a responsive design, a FoodiePress recipe editor, four theme designs and WooCommerce support.

2 ELANTRA 2015

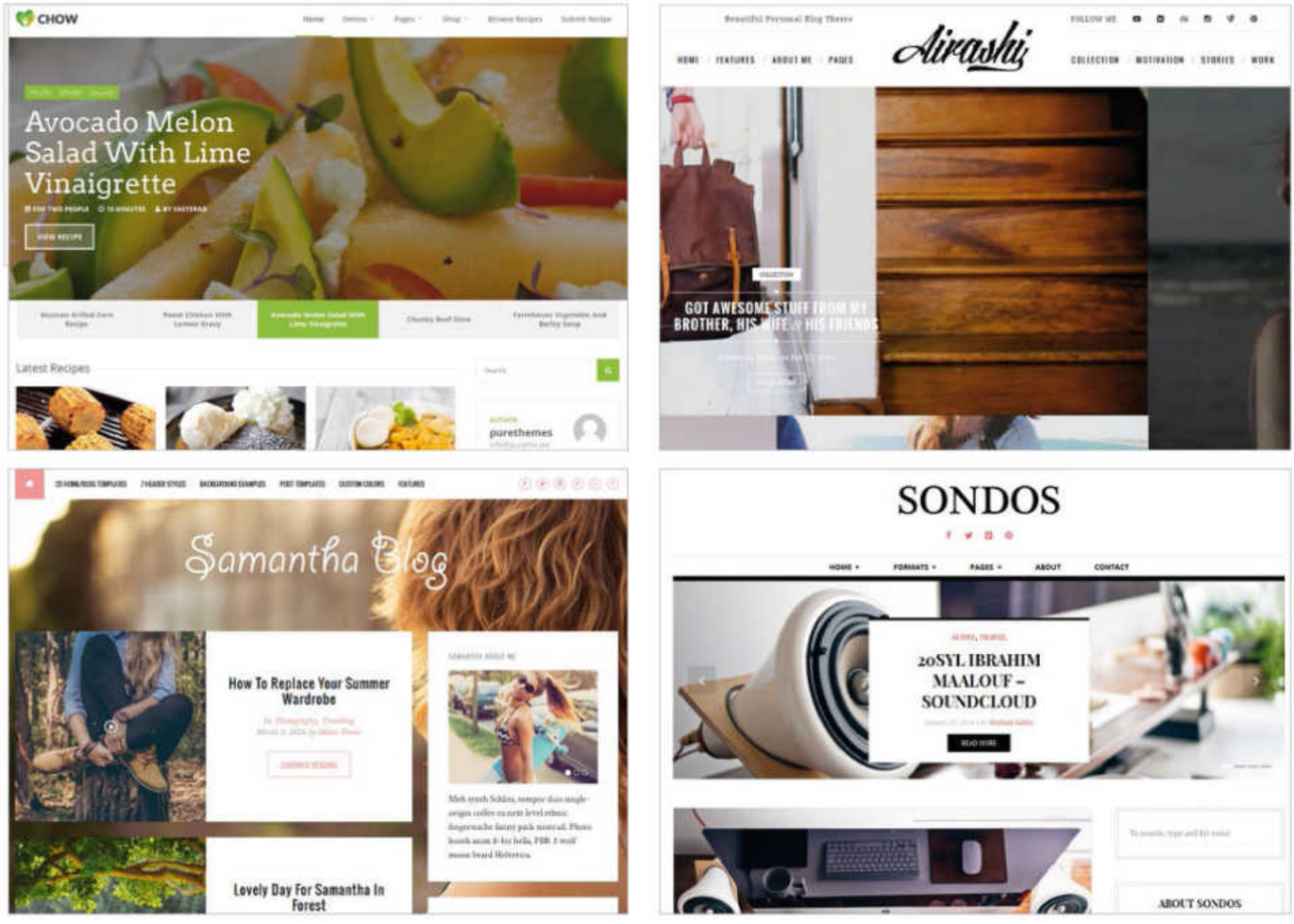
netm.ag/elantra-271

Built for enthusiastic bloggers, Elantra supports 10 post formats. You also get seven preloaded animations, infinite loading capability, WordPress 4.1.1 compatibility, several well-appointed homepage layouts and a fully responsive design.

3 F MAGAZINE

netm.ag/magazine-271

F Magazine is a simple, clean WordPress blog theme. It is responsive, Retina-ready and comes with an easy-to-use page composer. It can also be used to create an unlimited number of blog layouts.



Clockwise from top left Chow, Airashi, Samantha, Sondos

4 FASHIONISTAS

netm.ag/fashionistas-271

Fashionistas is perfect for fashion and photography. It has a clean, responsive masonry layout with a static header. It also offers unlimited hover colours and multiple post formats including standard, quote, link, image and gallery formats.

5 SAMANTHA

netm.ag/samantha-271

The responsive Samantha blog theme comes with 23 predefined page templates. It is also equipped with seven header styles, which will give you over 300 plus different homepage styles. What's more, the theme is responsive, with unlimited colour options.

6 SONDOS

netm.ag/sondos-271

Want a WordPress blog theme for photography, art or simply writing? Well, take a look at Sondos. It is a responsive and fully customisable theme that features seven post formats, three types of sidebars, six custom widgets and two archive templates.

7 LUXMAG

netm.ag/luxmag-271

The responsive LuxMag theme can be used for any subject matter. It comes with a custom search overlay, custom page templates, a theme customiser and built-in social sharing capabilities.

8 AIRASHI

netm.ag/airashi-271

Airashi has a strong aesthetic. It is Retina-ready, responsive and SEO-friendly, and offers three sidebar options and three header layouts.

9 ROSEMARY

netm.ag/rosemary-271

Rosemary is simple but elegant. It features five distinct blog layouts, four post formats, an Instagram feed, and a spectacular featured area slider.

10 MINIME

netm.ag/minime-271

miniMe's unique design is perfect for creating personal blogs, but can also be used for portfolio

and creative blogs. Portfolio post types supported include audio, video and slider.

11 MYBLOG

netm.ag/myblog-271

MyBlog is a premium blog theme that is perfect for anyone wishing to start a blog or a news site. It has a built-in admin panel and colour pickers, and two layout styles (blog or masonry).

12 AMALIE

netm.ag/amalie-271

Amalie is a premium blogging theme based on the Twenty Fifteen theme. You get a responsive design, the soliloquy slider, the WooCommerce plugin, and varied page templates.

13 VIXEN

netm.ag/vixen-271

Vixen is an elegant blog theme that can be used as full-width or boxed. It is responsive, SEO-friendly, and loads pretty quickly. Its minimal design also comes with five standard post types.

14 TWENTIES

netm.ag/twenties-271

This responsive premium theme comes with three colour skins. It is SEO-friendly, has a masonry grid, several post formats, and translation files.

15 PRESSMAN

netm.ag/pressman-271

This minimal theme is loaded with a variety of features including nine post formats, a Retina-ready, responsive design, support for all Google fonts, and unlimited colour options.

16 FRIDA

netm.ag/frida-271

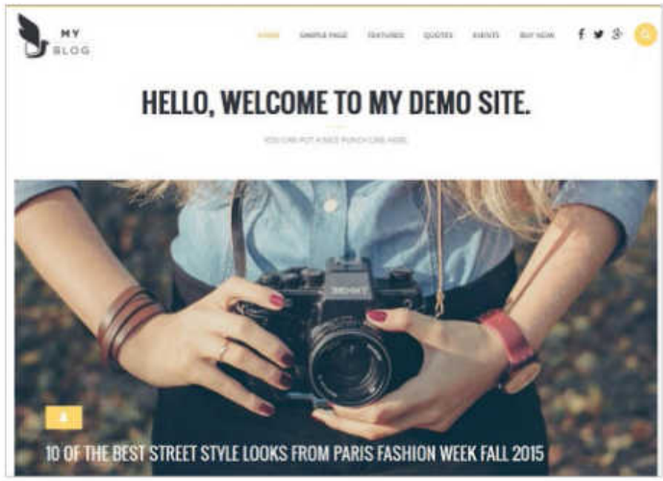
The creators of Frida refer to it as a sweet blog. It has five layout choices, four header options, support for 18 social accounts, and six custom widgets.

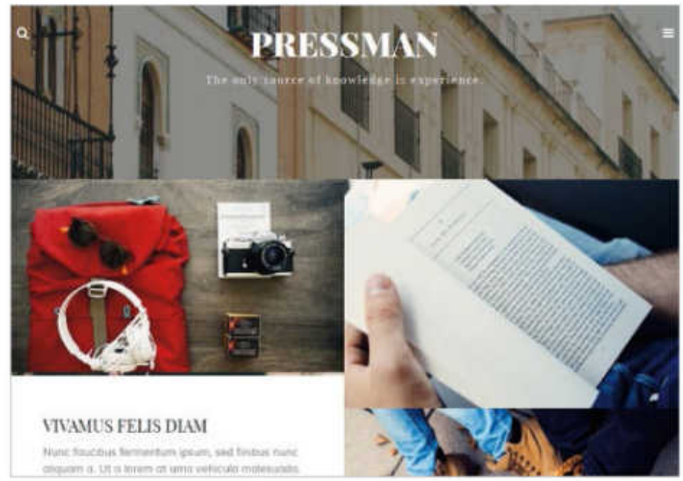
17 MAGNIFICENT

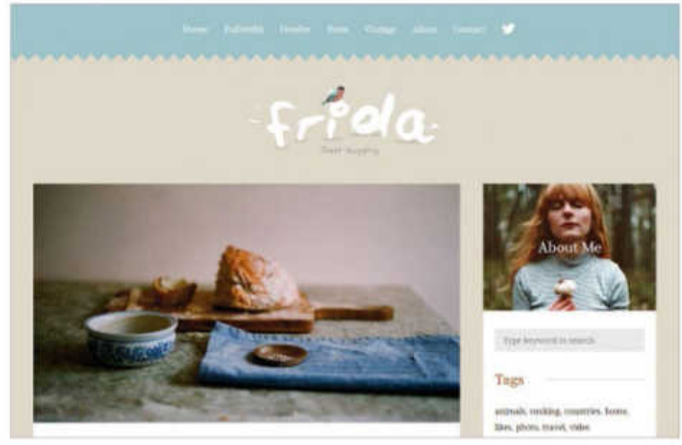
netm.ag/magnificent-271

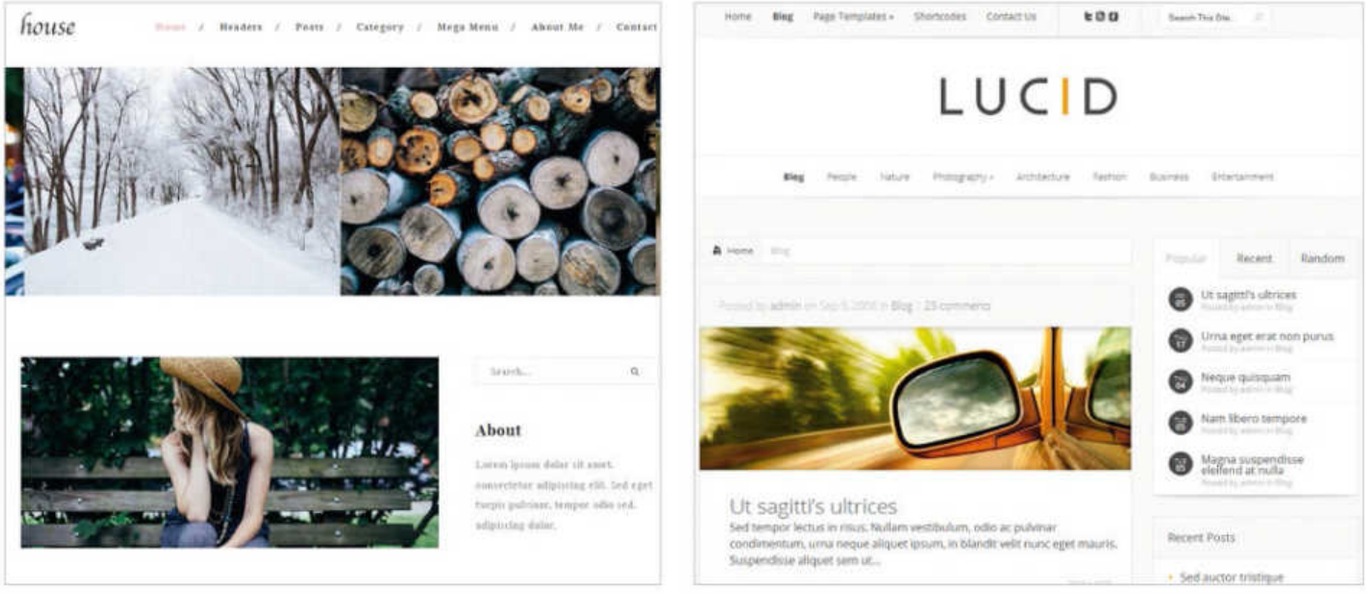
Magnificent is an elegant four-column WordPress blog and magazine theme. It is equipped with five

Clockwise from top left
MyBlog, Pressman, Vixen and Frida









Left to right House offers a variety of layout possibilities, and Lucid aims to retain an uncluttered feel

widget-ready areas and comes with seven predefined colours to suit your needs.

18 KROBS

netm.ag/krobs-271

Krobs has a clean and minimalist design. It has a touch-and-swipe design with fully responsive background imaging, and the option to use videos as backgrounds too. It also includes Retina-ready icons as well as over 600 Google webfonts.

19 BIRCH

netm.ag/birch-271

Birch's core strengths are simplicity and flexibility. This clean and responsive blog comes with over 600 Google fonts, a boxed and full-width layout, 23 custom shortcodes and over 30 different blog layouts. It is also HD/Retina-ready.

20 HOUSE

netm.ag/house-271

The House WordPress blog theme gives you a variety of different possibilities. It offers six post formats, three header styles, six blog layouts, a fully responsive design and isotope masonry. You can also customise its code if you wish.

21 LUCID

netm.ag/lucid-271

With the Lucid WordPress theme you can display as much content as you like while maintaining an uncluttered feel. It comes with five unique colours, a responsive design, secure code (ElegantThemes swears by that) and six page templates.

22 CEDAR

netm.ag/cedar-271

Cedar is a responsive theme that supports seven post layouts. It is optimised for fast loading, using CSS3 styling and a completely imageless layout. It comes with over 600 Google fonts.

23 SELFIE

netm.ag/selfie-271

Selfie, as its name suggests, is the perfect theme for a personal blog. It is built with the AJAX category filter and comes with unlimited colour variations.

Selfie, as its name suggests, is the perfect theme choice for a personal blog

24 NEXUS

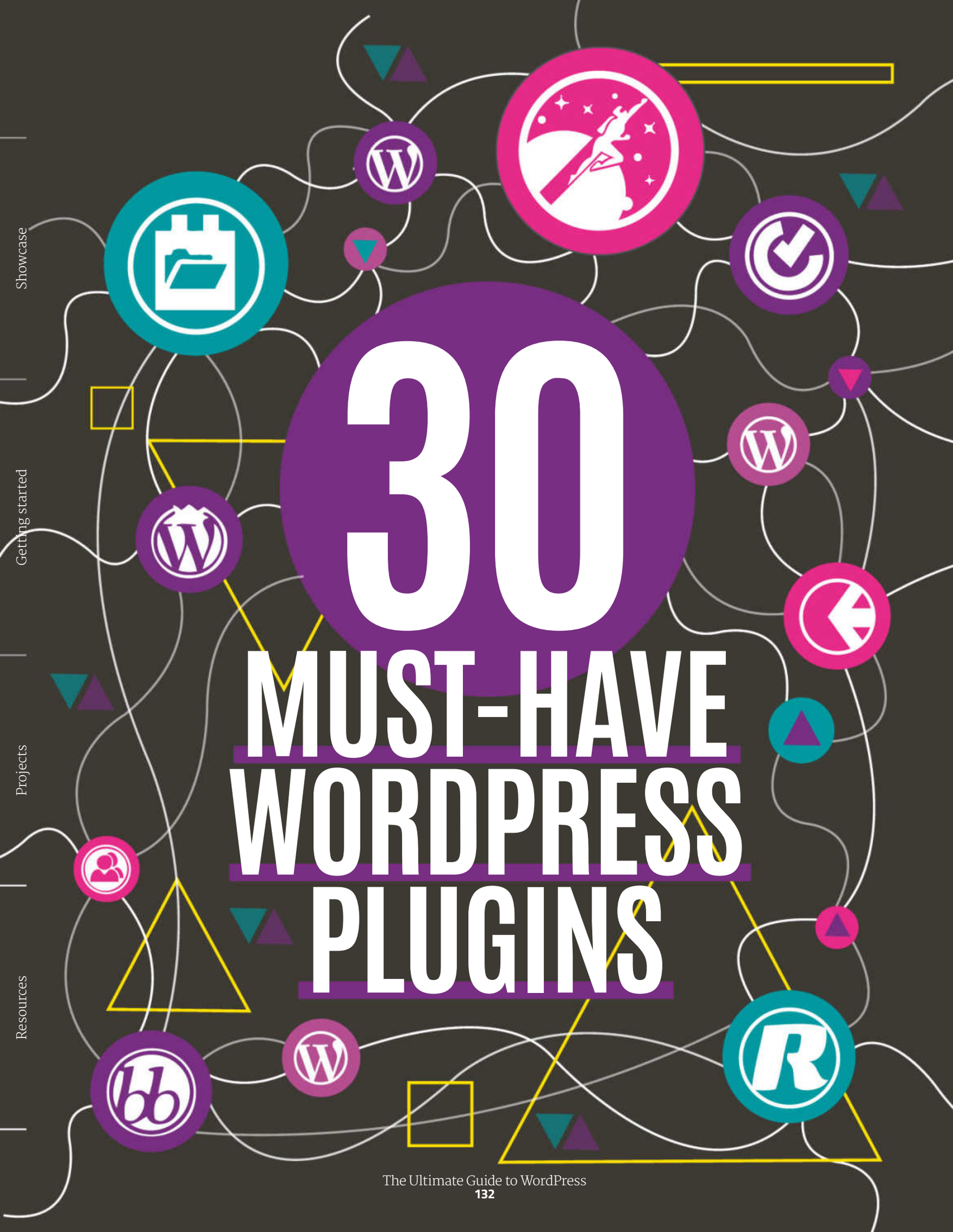
netm.ag/nexus-271

Nexus is a flexible blog theme from *ElegantThemes.com*. It offers varied shortcodes, six pre-styled page templates, and a homepage builder. It also gives you advertisement areas to monetise your blog.

25 BLOGOMA

netm.ag/blogoma-271

Blogoma is a responsive and Retina-ready premium theme. It has seven distinct post formats, unlimited colour options and nine built-in custom widgets. It is also SEO and translation-ready.



30 MUST-HAVE WORDPRESS PLUGINS

There are thousands of plugins in the WordPress directory, but which do you really need? **Shannon Smith** runs down 30 great plugins that will help make your life easier

With over 38,000 plugins in the official WordPress directory, it can be difficult to dig out the best ones. To help you I've put together a carefully curated list of the very best WordPress plugins available. These plugins will be especially useful for freelancers and developers. I chose options that might be helpful for building complex themes, plugins and custom websites that respect the latest web standards, and use the latest WordPress code base.

My list includes must-have plugins for spam control and SEO, there are great plugins for adding special types of content (like event lists and photo galleries), and I've included a special section on testing and debugging plugins that can be especially useful during development. I've also included some tools that help with version control, structured data and multilingual sites.

Most of the plugins on the list are freely available in the WordPress plugin directory (*wordpress.org/plugins*), but some are distributed independently and involve fees for upgrades and support. I also tried to choose plugins that respect the open source license (GPLv2 or later) used by the WordPress project. Let's dive in ...

SECURITY, PERFORMANCE AND BACKUP

01 WP SUPER CACHE

wordpress.org/plugins/wp-super-cache

WP Super Cache is designed to speed up your site. It takes the dynamic PHP files from your site and generates static HTML files, which it serves to visitors. This way, there are fewer calls to the database, and the PHP is only executed on an occasional basis for each page of your site. Besides making your site faster, it also protects it from spikes in traffic.

02 IMSANITY

wordpress.org/plugins/imsanity

This plugin limits the size, dimensions and resolution of any photos uploaded to your site. It's a great tool for multi-author blogs, and for client sites. You can configure all the settings, and also bulk-resize photos that are already in the uploads folder. Best of all, it's invisible to the user – it doesn't add any extra steps when users add photos to your site.

03 AKISMET

akismet.com

Akismet screens your comments for spam. It comes bundled with WordPress, but you do need to activate it. It relies on community spam reporting to help it improve its filters, which is one of the ways it stays consistently ahead of the spammers. Akismet also has an API that allows other plugins to use its spam filtering abilities. It's made by the folk at Automattic (*automattic.com*), and runs as a service for several content management systems.

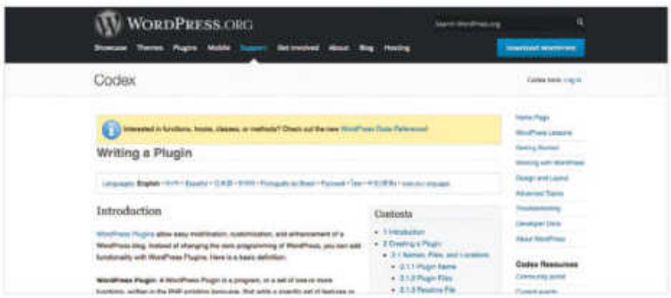
04 ITHEMES SECURITY

netm.ag/ithemes-271

Formerly Better WP Security, iThemes Security is another security tool, but this

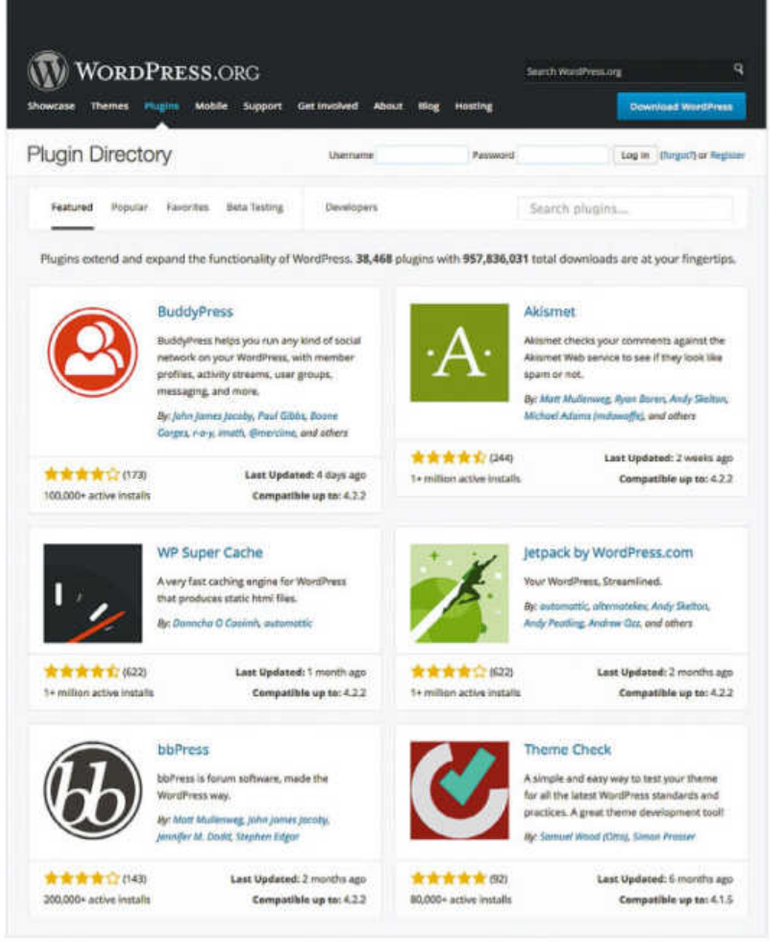
AUTHOR
SHANNON SMITH
Shannon is the founder of Café Noir Design (*cafenoirdesign.com*), a boutique web design company specialising in multilingual web development. She's interested in accessibility, open source software and sustainable practices

ILLUSTRATOR
BEX SHAW
Bex is a graphic designer and illustrator based in Bath. She is currently working as the art editor of **net** magazine
behance.net/bexshaw



Build your own On the WordPress codex you can find detailed instructions on the steps you need to follow to write your own, well-structured plugin

All in One You can use the All In One Schema.org Rich Snippets plugin to improve SEO, with the help of a very intuitive interface that's great for clients



Official directory The official WordPress plugin directory currently boasts over 38,000 plugins, and over 900 million total downloads

time it has some more tricks up its sleeve. The biggest benefit it offers is a security review of your site – it generates a list of problem areas that should be remedied. The list evaluates basic security holes, such as having a user with 'admin' as a username, or using the default database table prefixes. It also looks at whether or not strong passwords are being enforced, and if the uploads folder is executable. This plugin also lets you block your site to a range of IP addresses after a specified number of failed login attempts. It's a good way to protect yourself from both hackers and brute force attacks.

05 REVISR

wordpress.org/plugins/revisr

This newer plugin helps you use Git version control with your site. But unlike other Git plugins, Revisr lets you include your database. You can push and pull your files to services like GitHub, but you can also use it to create restore points for your database – something that even complex workflows don't allow. Revisr even lets you specify a development URL so that your database works on both your live and development site. Helps make version control automated and painless.

06 BACKUPBUDDY

ithemes.com/purchase/backupbuddy

BackupBuddy is a paid, commercial plugin. As the name implies, it enables you to back up your files and database, and restore either or both quickly if the need arises. All this through an easy-to-use, intuitive interface. Backups can be scheduled, and they can also be stored off-site using services like Dropbox or Amazon S3.

However, one of the most useful features offered by BackupBuddy is the ability to migrate your site between hosts with minimal effort. Unlike other migration workflows, this plugin takes into consideration serialised data and URL replacements. It also contains some handy server tools to help you with database repairs, Cron jobs and database search and replacement.

SEARCH ENGINE OPTIMISATION

07 ALL IN ONE SCHEMA.ORG RICH SNIPPETS

netm.ag/snippet-271

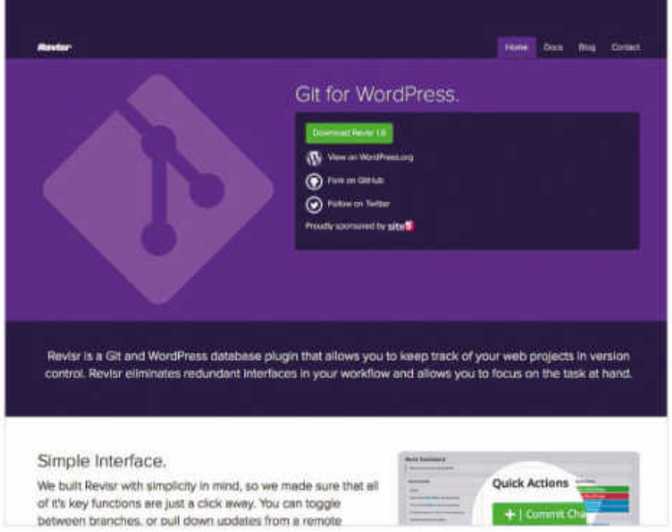
Structured data – also referred to as structured data markup or rich snippets – is one of the newest focuses of search engines (*developers.google.com/structured-data*). The All In One plugin lets you tell search engines not just what your data is about, but what kind of data you are offering. It currently includes rich snippet configuration for the following: reviews, events, people, products, recipes, video, software applications and articles, with more in development.

This plugin is a great way to get enhanced presentation in search results, but also to leverage some of the newer social media features in Twitter Cards and Facebook shares, for example.

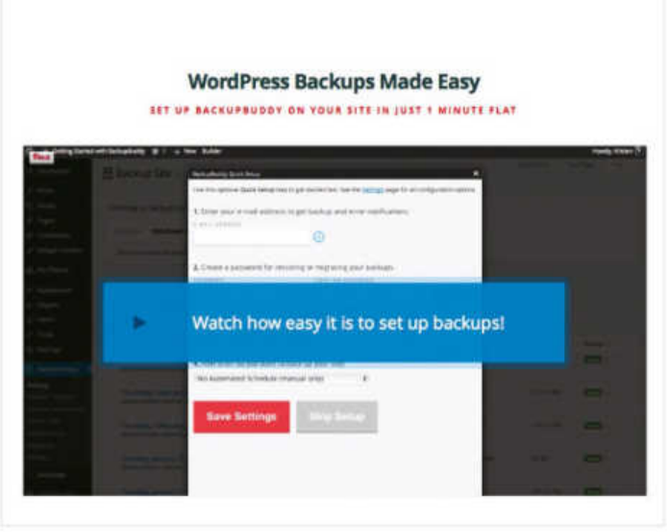
08 WORDPRESS SEO

wordpress.org/plugins/wordpress-seo

This search engine optimisation plugin comes from Yoast. It helps you determine how your site will show up in search results, as well as providing page analysis to help you keep your content focused, and creating XML sitemaps that search engines use to find all your pages and index your content. It also helps with social media integration and is multi-site compatible. Some of its less well-known features include the ability to add breadcrumbs to your site and optimise your RSS feed.



Revisr This simple plugin helps you manage your WordPress site with Git version control, inluding tracking changes and backing up your site



BackupBuddy This plugin lets you migrate sites between servers easily, while taking care of serialised data and URL replacements

If you already have a lot of content in your site, you can use the bulk editor to manage SEO features on a large number of pages at once. The developer behind the WordPress SEO plugin also provides a huge amount of easy-to-understand but detailed information on how search engines work on his site (*yoast.com/articles/wordpress-seo*).

POST TYPES, CUSTOM FIELDS AND WIDGETS

09 ADVANCED CUSTOM FIELDS

netm.ag/acf-271

This plugin lets you easily add metaboxes and custom fields to posts and pages. Once it's all set up, adding extra custom field information is easy – especially for clients.

10 TYPES

wordpress.org/plugins/types

Types manages custom fields too, but it also lets you manage custom post types, and custom taxonomies through a simple user interface. It's great for building complex sites with reduced code.

11 WIDGET LOGIC

wordpress.org/plugins/widget-logic

This is a useful tool for determining when widgets are active. It lets you use conditional tags to restrict where widgets are visible. It's especially helpful for complex site development.

SPECIALISED CONTENT

12 TABLEPRESS

wordpress.org/plugins/tablepress

TablePress lets you embed attractive tables into your site via a user-friendly interface. It also allows you to import and export files in Excel, CSV, HTML and JSON formats.

13 BLUBRRY POWERPRESS PODCASTING PLUGIN

wordpress.org/plugins/powerpress

If you want to add iTunes-compatible podcasts to your site, the Blubrry PowerPress Podcasting Plugin will do the heavy lifting for you. It uses Blubrry's platform for some features.

3 QUESTIONS TO ASK TO IDENTIFY A GOOD PLUGIN

There are three key questions you need to ask in order to identify a good-quality plugin. Let's take a look at these now.

1. Has the plugin been reviewed for security and quality?

As a general rule, the best place to get well-made WordPress plugins is the official WordPress directory (*wordpress.org/plugins*). As of 2011, new rules were implemented to make sure that the plugin directory is a better resource. For instance, plugins that haven't been updated in over two years are no longer shown in search results. Plugins are also reviewed for quality before they are accepted, and checked for malicious code, privacy issues, and 'advertising spam' (*netm.ag/quality-271*). The plugins here are free and fully functional (although often there are paid plugin upgrades available as well). Commercial plugin vendors do usually screen their plugins for security issues, but they are not nearly as transparent in publishing the standards they use.

2. Does the plugin respect the WordPress licensing terms?

WordPress is available for free to anyone under the GNU General Public License v2, or any newer version. It is also important to note that any software that contains elements that are derivative of WordPress's copyrighted code – including plugins – also need to be licensed under the same terms. That includes both free and commercial plugins. Good quality plugins are generally made by people who are happy to contribute to the WordPress open source project and respect its licensing terms.

3. Is the plugin well supported and compatible with the latest version of WP?

Within the WordPress plugin directory each plugin has its own public support forum, where you can see what problems (if any) users are having, and how much support the plugin developer is providing. There is also information on the latest compatible version of WordPress with each plugin, and easily available feedback from users on whether or not a given plugin works. The directory provides practical, transparent, community-contributed quality control. Many plugin developers also host their own forums, and those can also be a good resource, but not all are available for free.

NextGEN Gallery This helpful plugin lets you import all your photos as a single zip file

RESOURCES

Can't find a plugin that does what you want? Build your own!

- How to get your plugin in the WordPress plugin directory (*wordpress.org/plugins/about*)
- Detailed plugin guidelines (*netm.ag/pluginguide-271*)
- How to use subversion with the official plugin directory (*wordpress.org/plugins/about/svn*)

Don't want to start from scratch? Adopt an abandoned plugin!

Plugins looking to be adopted are tagged 'adopt-me' in the plugin directory.

- How to adopt abandoned plugins (*netm.ag/adopt-271*)
- Plugins looking for a developer to adopt them (*netm.ag/adopt2-271*)

Looking for some helpful tools? Try these

- Debug bar (*wordpress.org/plugins/debug-bar*)
- Query Monitor plugin (*wordpress.org/plugins/query-monitor*)
- Developer (*wordpress.org/plugins/developer*)

Need a tutorial? A step-by-step guide can help you get started

- Writing a plugin (*codex.wordpress.org/Writing_a_Plugin*)
- Beginner's guide to WP plugin development (*netm.ag/begin-271*)
- Anatomy of a WordPress plugin (*netm.ag/anatomy-271*)
- How to write a WordPress plugin (*netm.ag/write-271*)
- Build a WP contact form plugin in five minutes (*netm.ag/five-271*)

Learn about the plugin directory

- Behind the scenes in the WP plugin directory (*netm.ag/direct-271*)
- The WordPress plugin review team (*make.wordpress.org/plugins*)

14 NEXTGEN GALLERY

wordpress.org/plugins/nextgen-gallery

A full-featured image gallery tool, NextGEN Gallery lets you organise your photos and display them in a variety of styles and formats.

15 GRAVITY FORMS

gravityforms.com

Gravity Forms is a paid, commercial plugin that lets you add complex forms to your website and interact with a number of online services.

16 THE EVENTS CALENDAR

netm.ag/calendar-271

This plugin adds event listings to your website. It includes venue information, Google maps, and an events widget is on its way.

DEBUGGING AND DEVELOPMENT PLUGINS

17 THEME CHECK

wordpress.org/plugins/theme-check

This lets you check your theme against the latest theme review standards. It's a great development tool, especially if you're trying to get a theme into the official WordPress repository.

18 RTL TESTER

wordpress.org/plugins/rtl-tester

The RTL Tester plugin lets you test to see if your theme or plugin is compatible with languages that have a right to left (RTL) text direction. Another good theme development tool.

19 MONSTER WIDGET

wordpress.org/plugins/monster-widget

The Monster Widget plugin lets you easily test all the default widgets with your custom theme.

20 BROKEN LINK CHECKER

netm.ag/check-271

As the name suggests, this checks for broken links in your posts, pages, comments and custom fields. A good tool for making sure your site is production-ready.

21 DEBUG BAR

wordpress.org/plugins/debug-bar

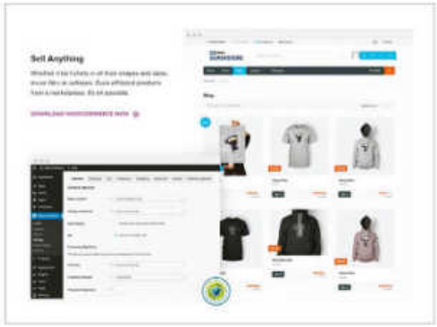
The Debug Bar plugin adds an extra menu to the admin bar that enables you to easily see information about PHP errors, SQL queries and calls to WP Query. It's intended to be used on staging sites to help developers write better, more compliant plugins and themes.

22 QUERY MONITOR

wordpress.org/plugins/query-monitor

If you are a developer working with transients, conditionals, hooks and filters, the Query Monitor plugin will help you identify what's going on behind the scenes.

Review and approve Plugins in the directory have been approved by the review team and rated by the community



Ecommerce WooCommerce powers 24 per cent of all ecommerce sites, and has been acquired by Automattic

Jetpack This plugin from Automattic ports the hosted WP network's features to self-hosted sites

23 LOG DEPRECATED NOTICES

netm.ag/log-271

The Log Deprecated Notices plugin helps theme and plugin developers identify the use of deprecated files, functions and function arguments. It's also useful for identifying appropriate alternatives.

24 REGENERATE THUMBNAILS

netm.ag/thumb-271

If you've ever developed a site then changed themes, or modified the default image dimensions on a custom theme, then the utility of this plugin will be immediately useful. Regenerate Thumbnails lets you resize all your featured images and thumbnails (or a subset) in one step.

25 BETTER SEARCH REPLACE

netm.ag/search-271

The Better Search Replace plugin lets you find text strings in your database and replace them at once – a very useful development tool with a simple, intuitive interface that lets you manage your database with ease.

26 DEVELOPER

wordpress.org/plugins/developer

Developer is a master plugin. It helps you assemble all the plugins and tools you need for theme or plugin development, so you can optimise your development environment.

COMPLEX PLUGINS

27 WOOCOMMERCE

woothemes.com/woocommerce

WooCommerce is one of the best ecommerce options around for a WordPress site, and one of the easiest to use. It's a full-feature ecommerce option and includes the following: multiple payment gateway extensions, inventory management, various shipping and tax configurations, coupon and marketing options.

It also manages its own repository of themes and plugins that integrate into WooCommerce and provide additional functionality. In May 2015, WooCommerce was acquired by Automattic, which should lead to an even tighter integration into WordPress in the future.

28 JETPACK

jetpack.me

Created by Automattic, Jetpack ports some of the best features of the hosted *WordPress.com* network to self-hosted sites. It is a large plugin, and contains a lot of features. Basic features include security improvements, statistics and great social sharing.

However, Jetpack has a lot of features that get a lot less attention, despite providing huge improvements. It allows you to have simple contact forms, better search functionality, and store intent on the *WordPress.com* content delivery network, which can greatly decrease your site's loading time.

29 EDIT FLOW

editflow.org

Edit Flow is an editorial calendar, a tool that helps individuals and publishing teams organise the publishing workflow. It includes a calendar, internal communication tools and notifications, and specialised user roles to help keep everyone organised. This plugin works with multi-site networks.

30 WPML

wpml.org

WPML is a paid, commercial plugin that allows you to operate a single website in multiple languages at once. It's one of the most complete multilingual solutions available and helps you translate themes and plugins, menus and widgets, as well as your content. It also integrates with some larger more complex plugins, like WooCommerce.

CONCLUSION

Every WordPress project is different, and no one plugin is a must-have for each one, but I hope with the aid of this list you'll start to recognise some of the better-supported, reliable standbys. With any luck, in this list you'll have discovered some new and useful tools that will make your life as a developer a little easier.

40 GREAT WORDPRESS TUTORIALS

We run down the resources you need to help you get started, supercharge your workflow, and become a WordPress professional

WordPress pro Rachel McCollin draws on her personal experiences in this article

Even though we've brought you a selection of expert WordPress tutorials in this special edition, we thought you may be hungry for more (of course you are!). So we've picked 40 brilliant WordPress tutorials from around the web that will boost your skills.

Because of its enormous popularity, there's a wealth of WordPress tutorials online to help you get to grips with the content management system – and here are some of the best from the last few months. Whether you're a beginner looking to get started or a current WordPress user aiming to further your skills, you'll find something here to help you ...

CAREER

01 The best places to find WordPress jobs and build your career

netm.ag/mccollin-271

In this post, Rachel McCollin helps you find the best place to identify your next WordPress opportunity, with a run-down of 13 sites you can use to find a WordPress job.

02 How to become a WordPress professional

netm.ag/mccollin2-271

How do you go about becoming a WordPress professional? In this post Rachel McCollin attempts to answer that very question. She also provides some useful tips based on her experience of forging a successful career with WordPress, as well as the experiences of other people she's spoken to.

GETTING STARTED

03 Toolbox of the smart WordPress developer

netm.ag/toolbox-271

This series on Tuts+ looks at several great tools and utilities that we can use with our WordPress websites. It covers plugins, themes and other scripts that will help you build better websites with WordPress.

04 Fifty actions of WordPress

netm.ag/actions-271

This series of tutorials on Tuts+ explores actions – one of the two different kinds of WordPress

hooks the application offers that enable users to customise execution.

05 The Tuts+ guide to template tags

netm.ag/tags-271

This series covers one of the fundamental concepts of WordPress: template tags. After a quick introduction, the series runs through four batches of almost 200 template tags.

06 Beginner's guide to PHP for WordPress

netm.ag/php-271

When you first get into customising WordPress sites, you quickly realise that a lot of WordPress relies on PHP, the server-side programming language. This three-part series of articles – an accompaniment to the Treehouse course on PHP for WordPress – covers some important things to remember when you start trying to customise PHP files and PHP code in WordPress.

07 Back to basics with WordPress CSS: Understanding the native classes

netm.ag/basics-271

In this guide, Karol K has a quick look at the absolute minimal set of CSS you need to take care of when building a WordPress site.

08 WordPress hooks: Actions, filters and examples

netm.ag/hooks-271

Read this article on Treehouse to discover exactly what WordPress hooks are, find out more about the different types of hooks, and look at a few examples of hooks in action.

GOING FURTHER

09 Rapid frontend prototyping with WordPress

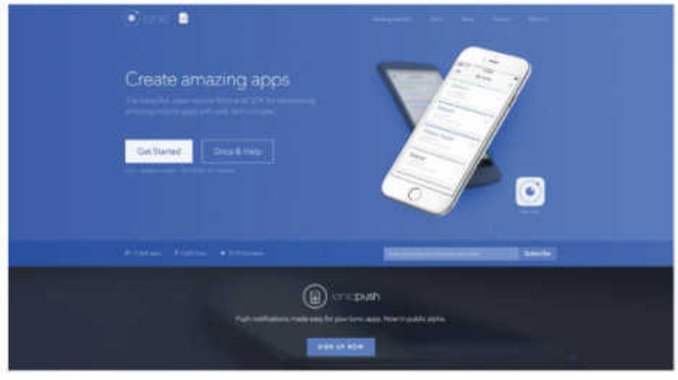
netm.ag/pataki-271

In this article, Daniel Pataki shows us that by using WordPress, highly interactive prototypes with great visuals are not at all that difficult to make. He looks at some basic prototyping concepts, then sets up a test server from scratch, converts an HTML template to a WordPress theme and explains how to use its items to create what we need.

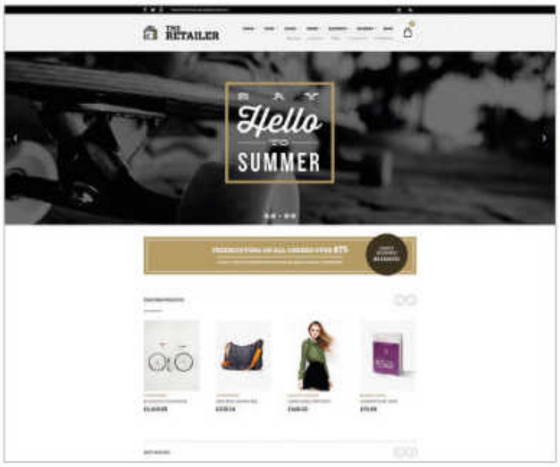
10 What to consider when choosing a WordPress theme

netm.ag/taylor-271

The choice of themes on offer can be overwhelming. In this post, Marcus Taylor shares what he believes are the most important factors to consider, so you know exactly what to bear in mind the next time you're on the hunt for a good theme.



Clockwise from top left Your guide to understanding the native classes; WordPress hooks enable your plugin to 'hook' into the rest of WP; Learn how to master rapid frontend prototyping; With Ionic, you can build a mobile app that uses WP data



Theme options There are several theme options for you to choose from

11 How to create better, more accessible WordPress themes

netm.ag/access-271

It is our responsibility as creators of WordPress themes to make them accessible to all users on any device. This article offers some simple tips to create better, more accessible themes.

12 How to create a mobile-ready site

netm.ag/mobile-271

This article covers how to build or buy a responsive WP theme. It also lists a few budget-friendly themes that are as responsive as they are beautiful, as well as some mobile plugins will help you to create a mobile-ready WordPress site easily and quickly.

13 9 lessons learned while creating a WordPress app

netm.ag/lessons-271

Slocum Studio recently created a WordPress app for one of its clients, so the team decided to share their knowledge on minimum viable products, user experience, responsive viewports and more.

14 Build a mobile app with the WordPress REST API and Ionic

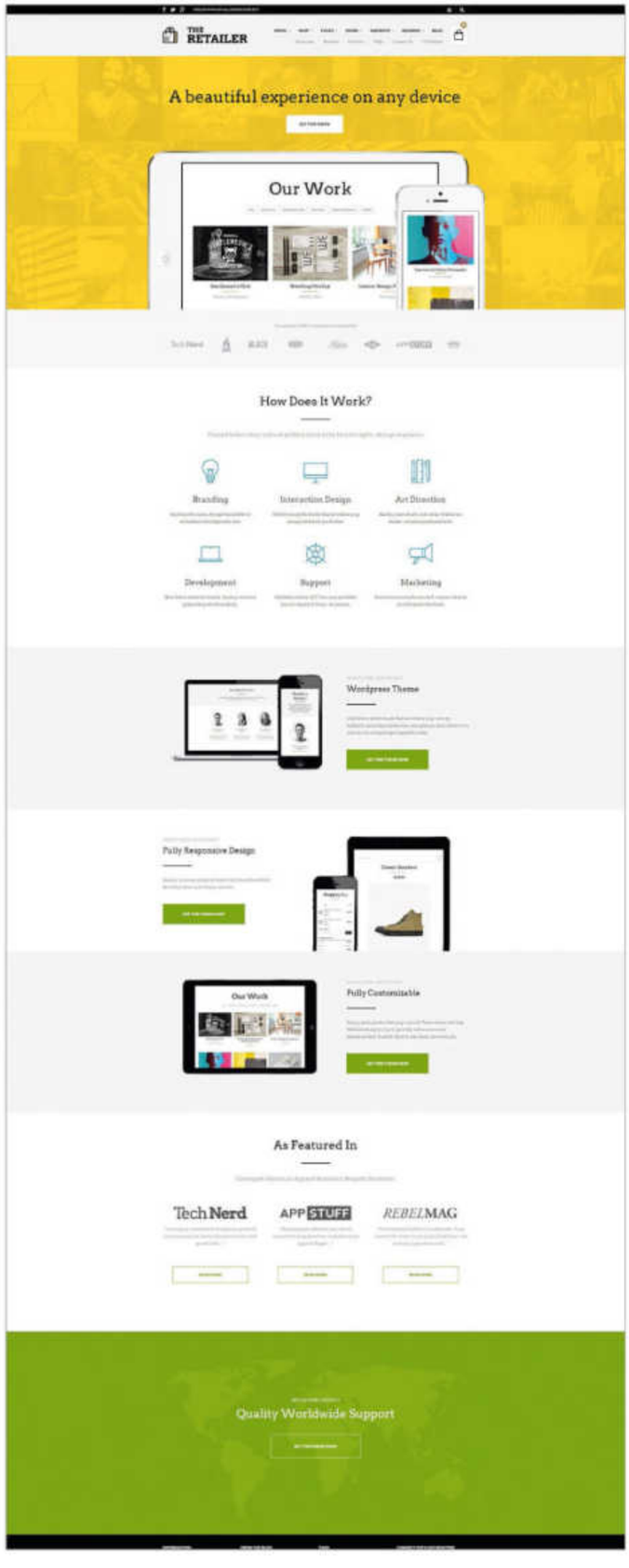
netm.ag/ionic-271

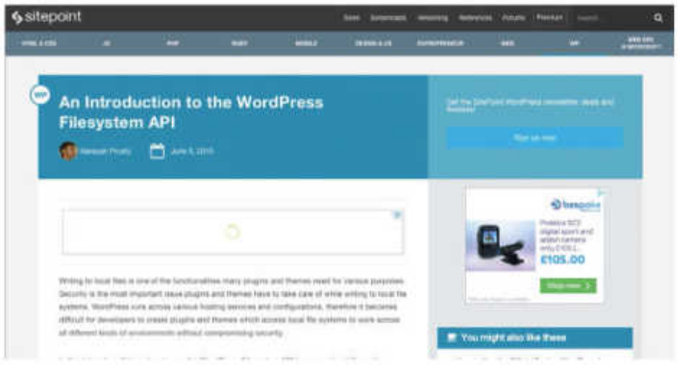
Learn how to get WordPress content into mobile apps using the REST API and Ionic, a mobile UI framework that helps people build awesome mobile apps easily. In this article you'll work with Ionic to build a mobile app that uses WordPress data.

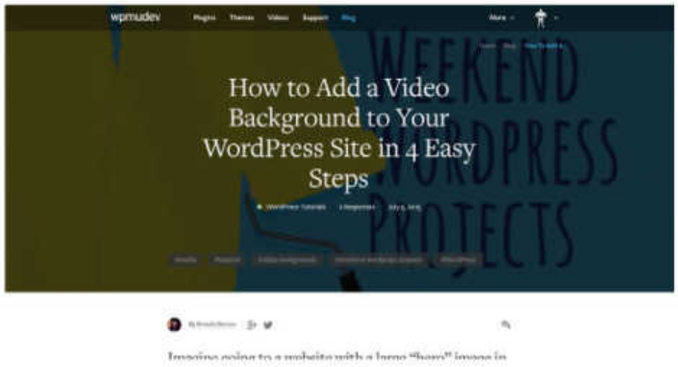
15 Extending WP with custom content types

netm.ag/custom-271

WordPress can be easily extended to fit the requirements of a custom data architecture.







Going further Take your website to the next level with a useful API, by adding an interesting design feature or by optimising your images

This article explores the process of registering new data types in a fully compliant manner.

16 A detailed guide to WordPress custom page templates

netm.ag/template-271

Among the most important tools in the quest for complete website control are page templates. They allow users to dramatically alter their website's design and functionality. Read this article to find out exactly how.

17 A walkthrough on conditional tags in WordPress

netm.ag/conditional-271

One of the most important strengths of WordPress is its extensibility. To create such a solid infrastructure, WP includes lots of handy sub-systems. One of them is Conditional Tags, which allow our code to function differently in particular situations.

18 Building an advanced notification system for WordPress

netm.ag/notification-271

An email notification system might be the best option if you want to promptly reach a target group. This article takes a closer look at how to let your website's subscribers decide when they want notifications.

19 How to completely customise the WordPress login page

netm.ag/login-271

Customising the WordPress login page can get a little awkward. In this tutorial Raelene Wilson shows you how to get started modifying the login screen so you can make it to look exactly how we want.

One of the key tools in the quest for complete website control are page templates

20 An introduction to the WordPress Filesystem API

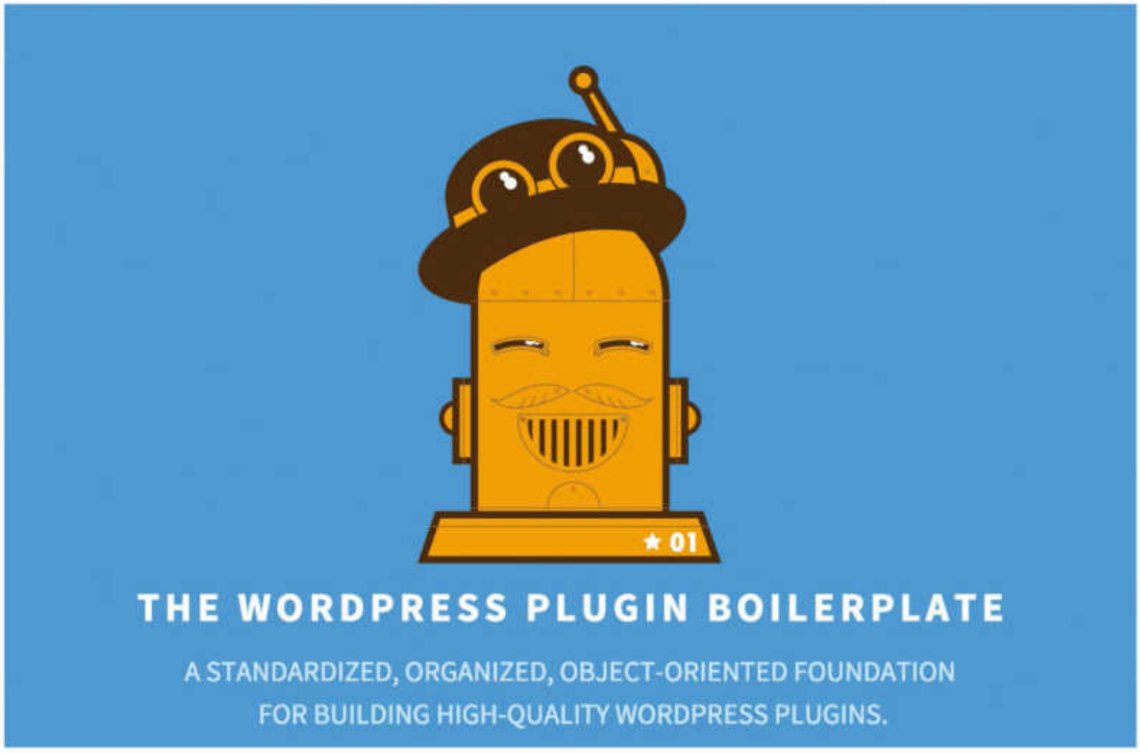
netm.ag/filesystem-271

This tutorial covers how to use the WordPress Filesystem API to access local file systems that take care of file permissions. You'll create a plugin that displays a form with a text area, in an admin page that saves the content of the text area to a file.

21 Using the Google Maps API and WordPress

netm.ag/maps-271

Over the course of a few posts, Tom McFarlin looks at some of the ways you may want to employ the Google



Plugin boilerplate
The WordPress plugin boilerplate follows coding and documentation standards and helps lay the foundation for your projects

Maps API in your project, what it entails, and how to get started with it.

22 How to add a video background to your WordPress site in four easy steps

netm.ag/video-271

Adding a video background to your WordPress website doesn't have to be daunting. Here's a look at just how simple it can be: just follow these four easy steps to add a video background, fast.

PERFORMANCE AND WORKFLOW

23 How to make your WordPress site blazing fast

netm.ag/blazing-271

This article focuses on what you should be doing to make sure your WordPress sites are as fast as possible. It covers frontend and backend performance, including CSS, JavaScript, images, caching and gzip.

24 The complete guide to mastering image optimisation for WordPress

netm.ag/image-271

Optimising your images is one of the simplest and most effective steps you can take to improve your site. Neglecting this step can damage user experience significantly. This article goes deep on the subject.

25 RICG responsive images for WordPress

netm.ag/ricg-271

A great introduction by Tim Evko to the plugin that integrates responsive images into the WordPress platform, with absolutely no effort needed by the user. For more on this, check out 'Adding responsive images to advanced custom fields in WordPress' (*netm.ag/add-271*).

26 Speed up development using the WordPress plugin boilerplate

netm.ag/boil-271

The WordPress plugin boilerplate aims to provide a standardised, high-quality foundation upon which to build your next awesome plugin. This first part of the series takes a deep look into the boilerplate, including how the files and folders are structured, as well as the code organisation of the boilerplate.

27 How to better manage WordPress pages with nested pages

netm.ag/pages-271

Do you use pages in WordPress? If so, then you probably understand how frustrating it is to manage a site with a lot of WordPress pages. That's where nested pages come in, and this article explains how to use them.

28 Using gulp for WordPress automation
netm.ag/auto-271

In this article you will learn how you can use gulp to automate several tasks that you usually do by hand, or for which you rely on tools developed by others.

29 How to install WordPress locally with Vagrant
netm.ag/vagrant-271

Developing locally is one of the best things you can do. Not only does it let you dispense with upload/download times, you can create as many projects as you want, work with real domains locally and generally speed up everything you do. This article shows you to do it with Vagrant.

30 VersionPress – WordPress meets version control
netm.ag/version-271

With the increasing popularity of version control, most developers are now accustomed to its capabilities. This article focuses on VersionPress, a plugin that aims to apply version control to a WordPress project.

31 Moving to HTTPS on WordPress
netm.ag/https-271

Chris Coyier recently took CSS-Tricks 'HTTPS everywhere'. That is, every URL on that site enforces the HTTPS (SSL) protocol. Non-secure HTTP requests get redirected to HTTPS. Here's some notes from his journey.

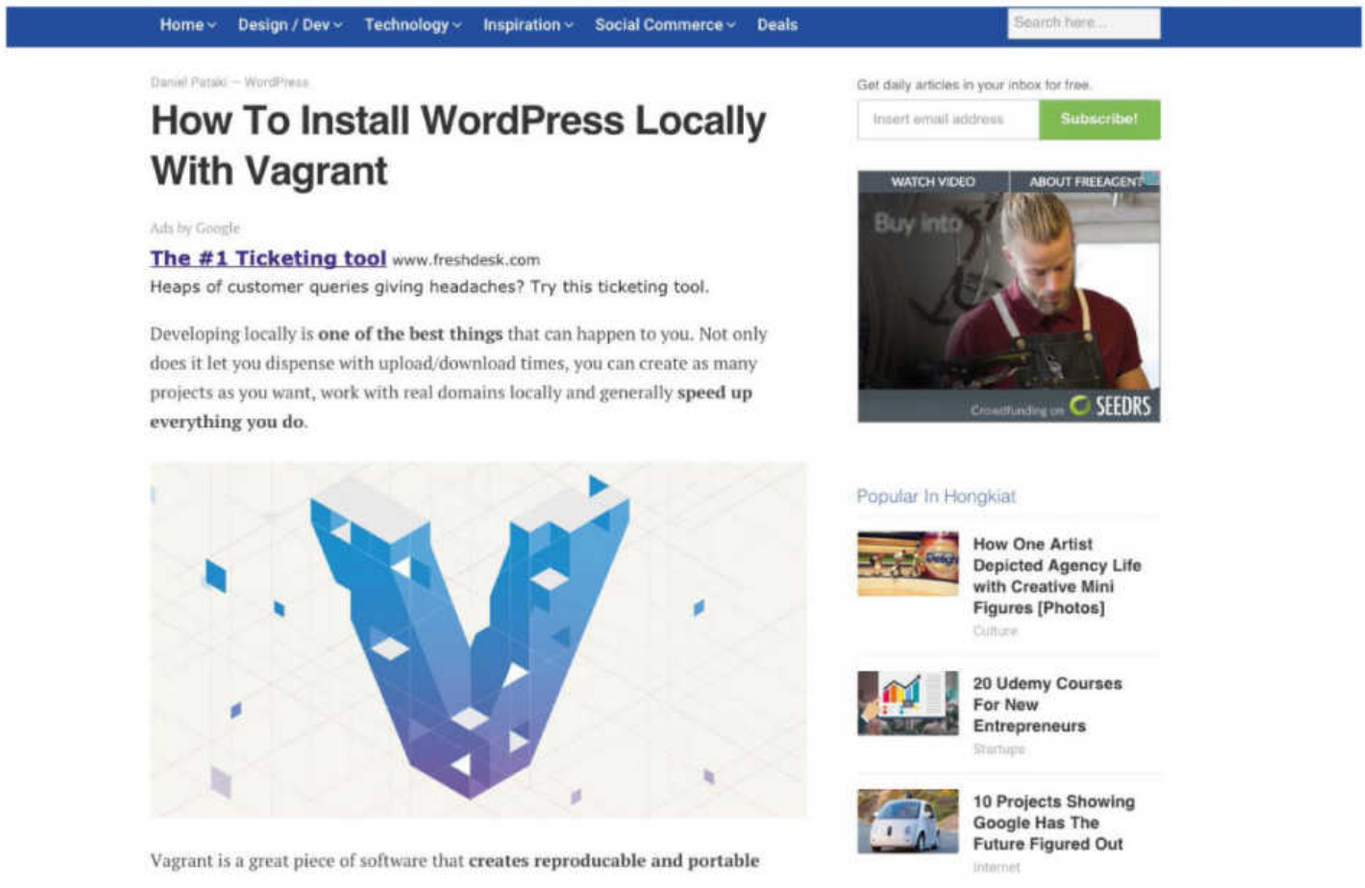
SECURITY AND THE WORDPRESS DATABASE

32 Introduction to WordPress frontend security: Escaping the things
netm.ag/thing-271

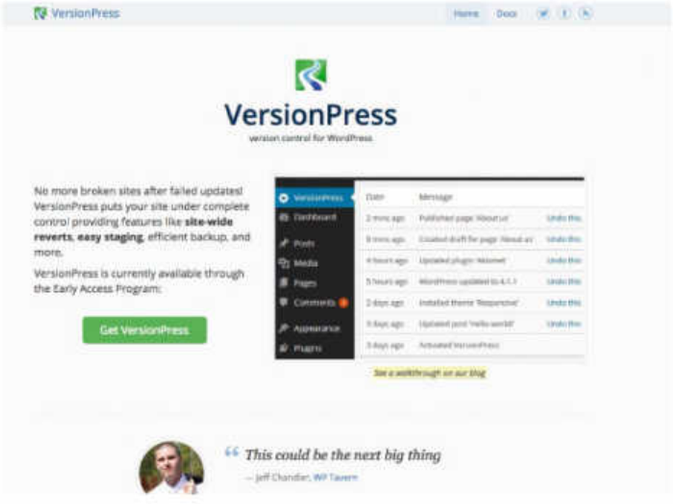
WordPress gives you all the tools you need to make your theme or plugin secure. You just need to know how and when to use each tool. Read this article to find out.

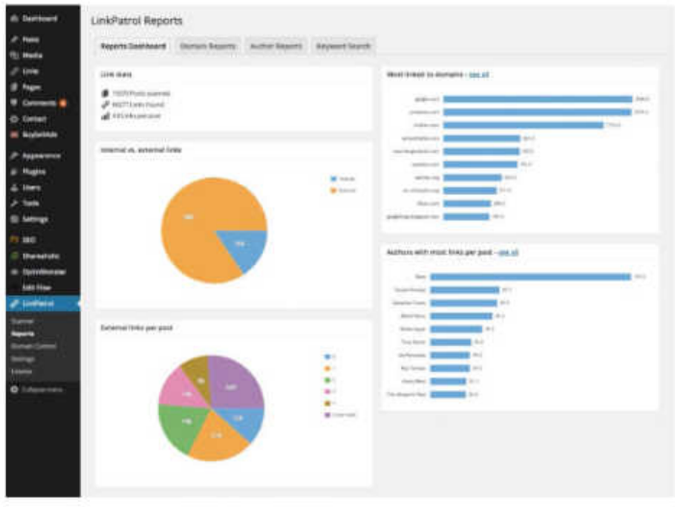
33 10 tips to secure WordPress
netm.ag/secure-271

This article looks at 10 ways you can improve and maintain your WordPress site's security, ranging from basic user authentication through to coding practices and hosting setup. For more



Vagrant Developing locally can speed up your workflow considerably





Left VersionPress is a plugin that aims to apply version control to WP projects

Right Managing broken links is key to providing a good user experience

advanced users, there are also some complex hardening techniques.

34 The developer's guide to the WordPress database (and writing better code)

netm.ag/database-271

The WordPress database may seem like a necessary evil to many developers, yet understanding how it works is a great way to start writing better code. This article sheds some light on each table in the database, what you can expect to find there, how functions affect it and what the bare essentials are.

35 How to back up your WordPress database

netm.ag/back-271

WordPress stores a ton of important stuff in the database. This article walks through some of the ways you can back up your WordPress database.

36 10 tips for keeping a squeaky clean WordPress (and multisite!) database

netm.ag/squeak-271

If you've been using WordPress for a while, chances are your site is due for a clean up. In this post, you'll learn several ways you can keep your database squeaky clean, with simple SQL queries as well as plugins. There are also tips for single and multisite installations.

HOSTING, MANAGING & MARKETING

37 Convert a website from any other platform to WordPress in 10 simple steps

netm.ag/platform-271

You started your website on a platform that was popular at the time. Now you've used it for a while, you've decided it no longer does what you need it to. You want to move your website to the mighty WordPress. This article shows you precisely how to make the shift, and either keep the structure you have or improve it.

38 The ultimate guide to choosing a WordPress host

netm.ag/host-271

These days you have an awful lot of options for hosting your website – so many that it's easy to get lost. This article helps you make a decision.

In this post, you'll learn several ways you can keep your database squeaky clean

39 Managing broken links and 404s

netm.ag/links-271

The management of broken links is an integral part of good WordPress maintenance. And, thanks to a number of plugins and tools at our disposal, it's becoming increasingly easy to automate the process of link maintenance these days. This article takes a look at some of these tools.

40 Three ways to better market your WP site

netm.ag/market-271

Once you have a WordPress site ready for launch, there is a lot you can do to help get it off the ground and then continue to grow. This article recommends plugins that will help you with marketing your WordPress site.

ACCESSIBILITY

UX AND WORDPRESS

Joseph Karr O'Connor explains how WordPress is working to improve the accessibility of its themes and core

There are three main components in a WordPress site: the core that runs the site via a database with controls provided through admin screens; plugins to extend features; and themes to change the look and functionality of a site. The WP Accessibility Team (*netm.ag/WPaccess-267*) is working on improving accessibility of themes and the core.

ACCESSIBILITY-READY THEMES

Laura Legendary sells braille jewellery using WordPress at *elegantinsightsjewelry.com*. Like many of her customers, Laura is blind. Before she started the site in 2009, there were no accessible commercial themes in the WP Themes Directory (*wordpress.org/themes*), so Legendary worked with a developer to make an existing theme (Weaver II) accessible enough for her purposes.

Last year, the Accessibility Team created a set of accessibility guidelines (*netm.ag/guidelines-267*) for developers. There is now the option of a new tag: accessibility-ready (so named because although a theme might be accessible when installed, the site owner and content providers are responsible for using accessible techniques for uploading content). There is documentation to help WP theme reviewers through the accessibility-check process (*netm.ag/review-267*). On my site I have a tutorial for adding accessible content to WordPress sites, plus a list of accessibility checking tools (*accessiblejoe.com/tools*). Today there are 37 accessibility-ready themes in the WP Themes Directory.

ACCESSIBILITY OF CORE

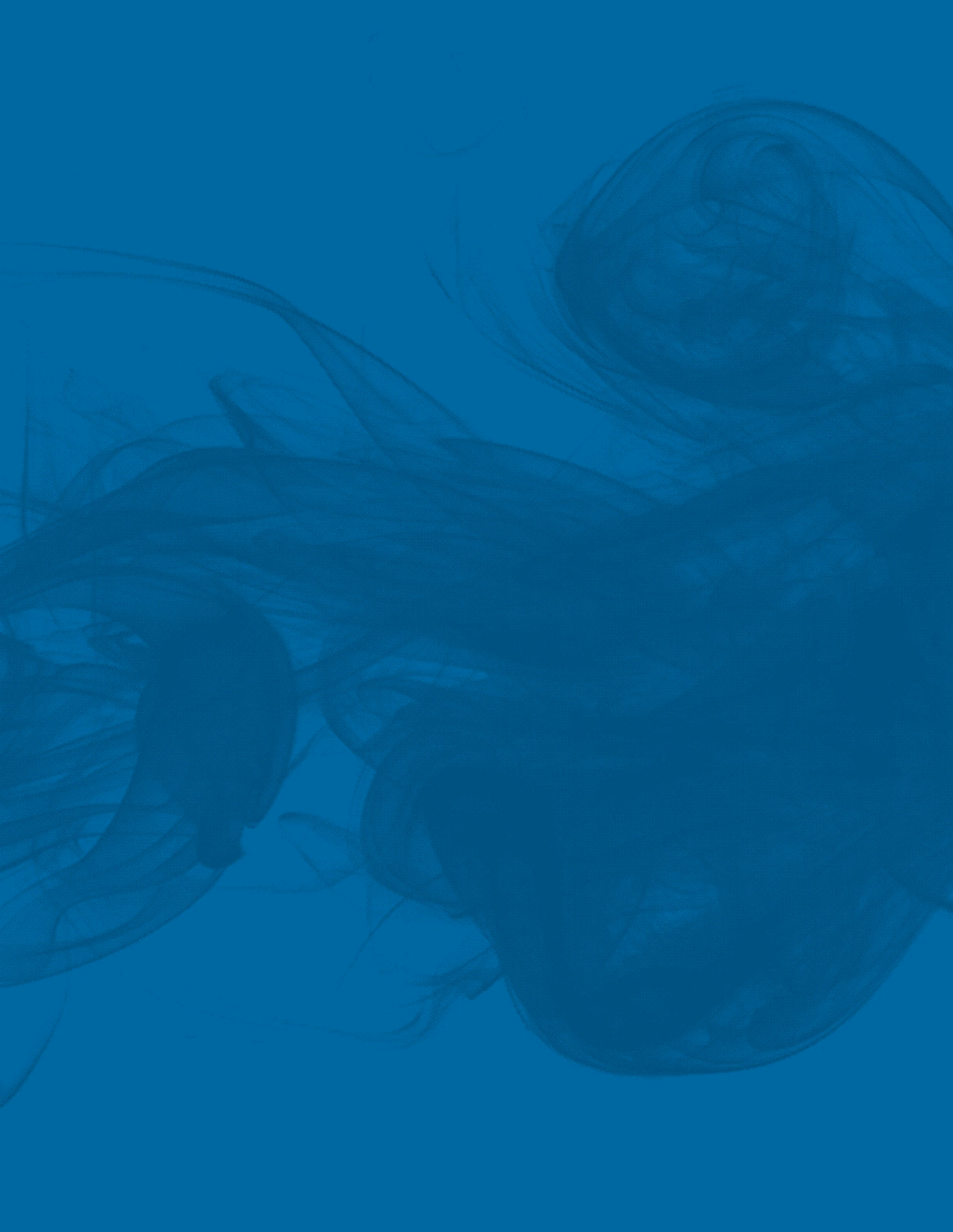
The WordPress Accessibility Team is also focused on core development. We are working through many issues affecting use of the admin screens. An excellent example is the accessibility test results of the soon-to-be-released Customizer Theme Switcher. This detailed report encompasses the true spirit of Accessible UX (*netm.ag/switcher-267*). Right now we are developing pattern libraries and improved documentation to provide scaffolding for core developers who are engaged and eager to learn.

THE TIPPING POINT

I believe the practice of accessible UX on WordPress has reached a tipping point. Conversations about accessibility now include many people who are new to the topic and want more information. As more countries around the world step up their own efforts, we will continue to improve WordPress accessibility to meet the challenge.

PROFILE

Joseph (@AccessibleJoe) has been an accessible UX practitioner since 1999, and currently helps the WordPress Accessibility Team and works with clients who value accessibility



THE ULTIMATE GUIDE TO WORDPRESS VOLUME II

Written by the industry's leading experts, The Ultimate Guide to WordPress Vol II shows you how to get the best from the world's most popular content management system. Discover the top themes and plugins, be inspired by the greatest sites built on WordPress, learn how to craft perfect themes, how to make your themes responsive, switch from CSS to Sass, boost your workflow with the REST API and much, much more

Become a WordPress master today!